NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

A NOVEL PROJECT MANAGEMENT THEORY AND ITS APPLICABILITY

by

Abdulkerim Erguner

March 2008

Thesis Advisor:	John Osmundson
Co-Advisor:	Kadir Alpaslan Demir

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March 2008	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE A Novel Project Management Theory and Its Applicability	**5. FUNDING NUMBERS**
6. AUTHOR(S) Abdulkerim Erguner	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

Software Project Management is an emerging discipline. The software project manager's job comprises every aspect of the project from starting the project to closing out. Practitioner's of the discipline use several project management tools in managing diverse aspects of software projects. However, there is no existing management theory that combines different aspects of a software project and results in a complete picture.

This study discusses a theory and modeling language which combine several management aspects of software projects into concrete models to aid the software project manager. The mathematical relations and graphical models derived from the theory comprise entire entities and activities of a project determined by the project team and depict any kind of relationships between the entities and activities including stakeholders. The theory provides a mathematical model for software projects and the modeling language provides graphical models of software projects representing the mathematical model.

This study tests the theory and the modeling language in two case studies for applicability. The results indicate that the theory and the modeling language is applicable to real world projects and show promise to be a valuable software project management tool.

14. SUBJECT TERMS Software Project Management, Stakeholders, Life Cycle Methodologies, Modeling Management	**15. NUMBER OF PAGES** 113
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

THIS PAGE INTENTIONALLY LEFT BLANK

A NOVEL PROJECT MANAGEMENT THEORY AND ITS APPLICABILITY

Abdulkerim Erguner
1st Lieutenant, Turkish Air Force
B.S. Electronics Engineering, Turkish Air Force Academy, 1999

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
March 2008

Author: Abdulkerim Erguner

Approved by: John Osmundson
Thesis Advisor

Kadir Alpaslan Demir
Co-Advisor

Dan Boger
Chairman, Department Information Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Software Project Management is an emerging discipline. The software project manager's job comprises every aspect of the project from starting the project to closing out. Practitioner's of the discipline use several project management tools in managing diverse aspects of software projects. However, there is no existing management theory that combines different aspects of a software project and results in a complete picture.

This study discusses a theory and modeling language, which combine several management aspects of software projects into concrete models to aid the software project manager. The mathematical relations and graphical models derived from the theory comprise entire entities and activities of a project determined by the project team and depict any kind of relationships between the entities and activities including stakeholders. The theory provides a mathematical model for software projects and the modeling language provides graphical models of software projects representing the mathematical model.

This study tests the theory and the modeling language in two case studies for applicability. The results indicate that the theory and the modeling language is applicable to real world projects and show promise to be a valuable software project management tool.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

First, I would like to thank Professor John Osmundson and 1st Lt. Kadir Demir for their support and guidance during this thesis preparation.

Second, I would like to thank to my wife Zehra for her unconditional love, support and understanding throughout our entire stay here in Monterey.

Third, I would like to thank to my parents and brother for their support throughout my entire life.

Finally, I would like to thank the Turkish Air Force for giving me the opportunity to study at the Naval Postgraduate School.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

This thesis discusses a novel Software Project Management Theory and its applicability. The theory helps us to understand the static and dynamic aspects and interactions of project. It lays a foundation for a modeling language that aids us in building generic and custom models for project management. It also helps us in modeling the management of a software project graphically to aid managers and individuals participating in the project for planning, monitoring and execution purposes.

A. PROBLEM STATEMENT

Software project management is an emerging subject. Even though there are numerous studies and books on software project management; there is still a lack of a well-established project management theory. The body of knowledge consists of principles, practices, standards and frameworks.

A unifying theory and analysis of project managements in terms of formal methods should be the next step in the practice of software project management. The novel theory that this thesis discusses suggests that software project management can formally be explained and analyzed. The theory is supported with a modeling language. This thesis intends to explore this new theory and its applicability to the real world.

B. OBJECTIVES

The objective of this thesis is to test the applicability of a novel software project management theory and modeling language. The specific goals of this thesis include:

- To identify the need for a new theory and modeling language,
- To identify the required features for a modeling language,

- To apply the new theory and the modeling language to generic and real life examples in order to reveal the upsides and downsides of the theory,
- To recommend required modifications and additions to the theory and the modeling language.

C. RESEARCH QUESTIONS

The specific research questions addressed in this study include:

- Why there is a need for modeling the project management?
- What are the features of existing project management tools?
- What is absent in existing tools?
- What features should a project management modeling tool have?
- Is the novel software project management theory and the modeling language applicable to generic models and real world examples?

D. THESIS ORGANIZATION

Chapter II gives a background on software project management and discusses discuss projects, project management, project management related areas and proven lifecycle methodologies.

Chapter III will poses and answers some research questions on software project management.

Chapter IV explains the new theory and the modeling language.

Chapter IV also describes the features that the new theory.

Chapter V and VI describes the application of the new theory to two separate case studies. These chapters aim to test the applicability of the new theory to real life and generic models.

Chapter VII comprises the results of the thesis and recommendations for future studies

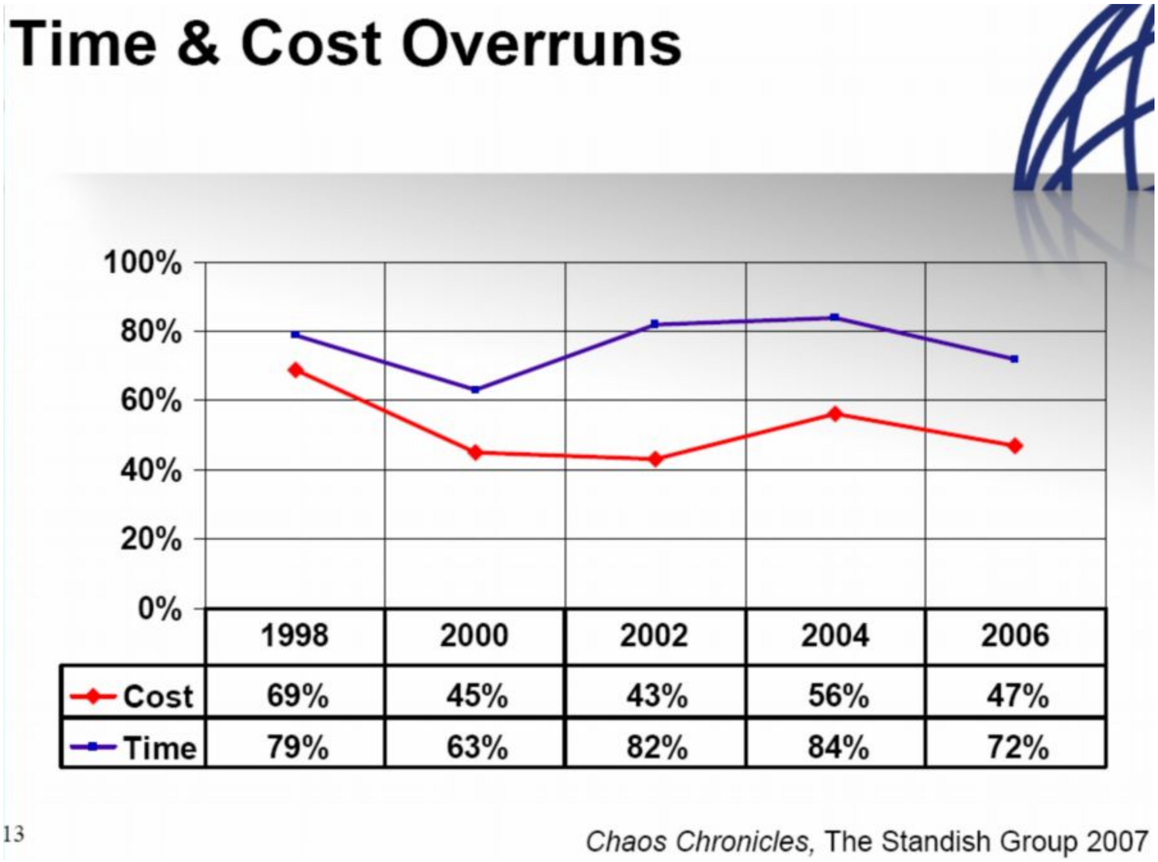
THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

A. WHAT IS A PROJECT

Project Management Institute’s PMBOK 2000 (Project Management Body of Knowledge) guide defines a project as “a temporary endeavor undertaken to create a unique product or service [1].” There are two key terms in this definition: temporary and unique. Temporary means that any project has a definite end. The project ends when desired objectives are reached or failure of the project becomes inevitable, or the organization’s need for the project no longer exists. Unique means the product has some distinguishing features differentiating it from other projects [1]. Therefore, “a project involves doing something that has not done before [1].”

“A project is a discrete set of activities performed in a logical sequence to attain a specific result. Each activity, and the entire project, has a start and stop date [2].” Research and reports suggest that, the planned schedule of software projects cannot be met in many cases. The Standish group Chaos report 2005 identifies that 84% of software projects have time overruns (Figure 1) [3]. A 2005 Standish group survey of 8,000 software projects shows that average project exceeded its schedule by 120% [4]. Therefore, planned dates should be flexible and reasonable. The project manager will have to revise schedule as project goes on. Then, we can say that a project should have a reasonable and flexible completion date for each activity and the entire project.


Time & Cost Overruns

	1998	2000	2002	2004	2006
Cost	69%	45%	43%	56%	47%
Time	79%	63%	82%	84%	72%

13

Chaos Chronicles, The Standish Group 2007

Figure 1. Project Schedule and Cost Overrruns (From: [3])

Projects vary in size and duration. They may involve a few people or thousands. They may last a few weeks to several years. Projects may involve a single unit of an organization or several units [1]. They may even involve several organizations and geographically dispersed locations over continents.

B. WHAT IS PROJECT MANAGEMENT?

PMBOOK 2000 defines Project Management as “the application of knowledge, skills, tools, techniques to project activities to meet project requirements [1].” Project management comprises the tools, techniques and processes to define, plan, organize, control, lead and close a project [2].

Project management is completing a project within defined scope, schedule, cost and desired quality. It involves managing people, resources, time, and budget. Software project management is specifically related to managing and controlling software product developments [5].

The project manager is responsible for planning, executing, controlling and staffing the project. He or she deals with anything related to project including ideas, things, and people [5].

Planning and organization of the staff has to be done in such a way that there is a confidence in the delivery of the final product within planned schedule, expected cost and quality. To accomplish this, the software project manager assembles a staff that is qualified enough to execute the required tasks. He or she has to provide necessary information and guidelines to the staff in executing the tasks. He or she has to put controls in place to prevent digression from the expected result of the tasks. Tomayko and Hallman define these controls as risk management [5].

Project management has several aspects that management of a project should address. PMBOK2000 defines these areas as project management related areas [1]:

1. Integration Management

Project integration management provides the coordination between various components of the software project to ensure that final product meets the expectations [1].

2. Scope Management

Scope management is defines the scale of the project based on the owner’s expectations, make sure that the project comprises what is needed, and the defined scope is not exceeded [1].

3. Time Management

Time management estimates, develops and controls the schedule of the project to ensure that the project is completed within the determined time scale [1].

4. Cost Management

Cost management comprises estimating, planning and controlling the cost of the project to make sure that the project does not exceed the approved budget [1].

5. Quality Management

Quality management controls and assures that the final product of the software project meets the expectations and needs of the stakeholders [1].

6. Human Resource Management

Human resource management comprises staff planning, acquisition and team development to make efficient use of individuals involved in the project [1].

7. Communications Management

Communication management comprises the processes that ensure proper and timely distribution of information throughout the life cycle of the project between stakeholders of the project [1].

8. Risk Management

Risk management comprises processes to identify, control and mitigate the risk for successful completion of the project. PMBOK identifies the components of risk management as risk management planning, risk identification, qualitative risk analysis, quantitative risk analysis, risk response planning, and risk monitoring and control [1].

9. Procurement Management

Procurement management comprises the processes for acquisition of supplies and services required for the project. PMBOK defines the components of procurement management as procurement planning, solicitation planning, solicitation, source selection, contract administration, and contract closeout [1].

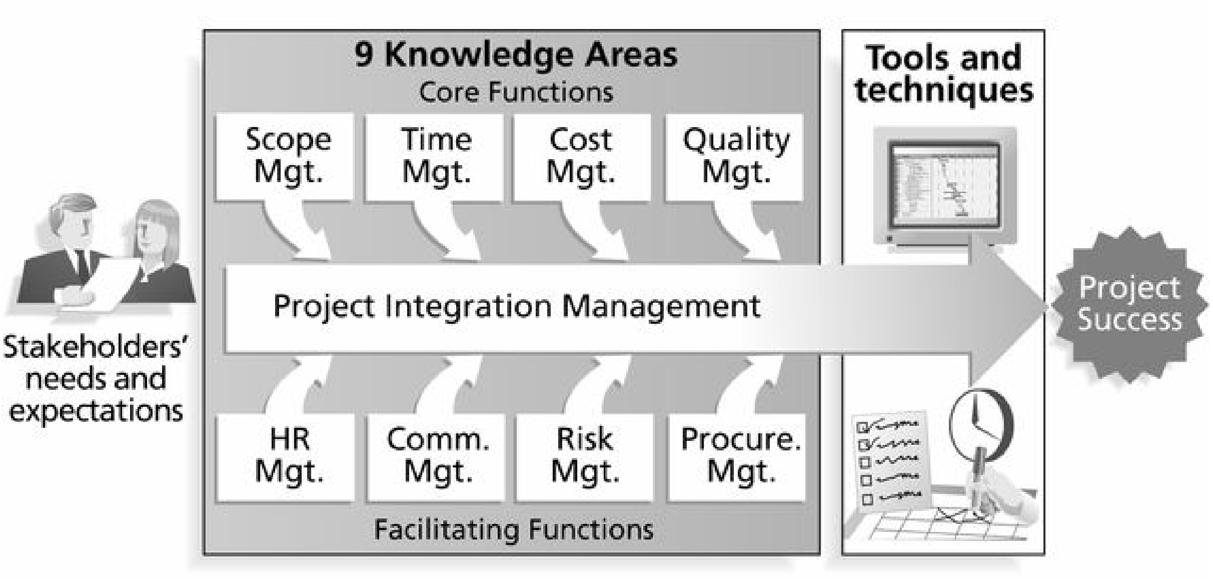


Figure 2. Project Management Knowledge Areas (From:[6])

As Figure 2 depicts, project success is closely related to effectively integrating project management knowledge areas based on stakeholders' needs and expectations.

C. PROJECT MANAGEMENT LIFECYCLE METHODOLOGIES

Projects are composed of tasks, subtasks and phases to ease the control and management. The phases of a project are accomplished in a predetermined sequence and have relations to each other and other aspects of the project like stakeholders, entities, etc. PMBOK2000 defines those phases as the project life cycle [1].

Each phase of a project results with a deliverable that is an input to the next phase. Deliverables are tangible entities that are attained by completion of any phase [1]. It may be a statement of requirements, a prototype, etc. Final deliverable for the project is the final product.

Project life cycle defines the entire project and phases. There are widely used methodologies used to define project phases and help in planning, controlling and execution purposes. Different methodologies can be used for different projects, or a combined methodology can be applied [6]. Features of the project - like requirements, scope – will determine which life cycle model to use. Following are some popular software lifecycle methodologies used in managing software projects:

1. The Waterfall Model

"The Waterfall model is based on iterative relationship between successive development phases [7]." In the Waterfall model, completion of one phase triggers another and those phases do not overlap. All planning is done upfront and phases are executed based on the original plan. Integration and testing occurs at the end, which is the first chance for everybody to see the final product. Waterfall methodology is good for projects that are quality oriented rather than cost and schedule. It is also good for projects that are based on well-understood requirements and technologies, and where final product definitions are stable [6].

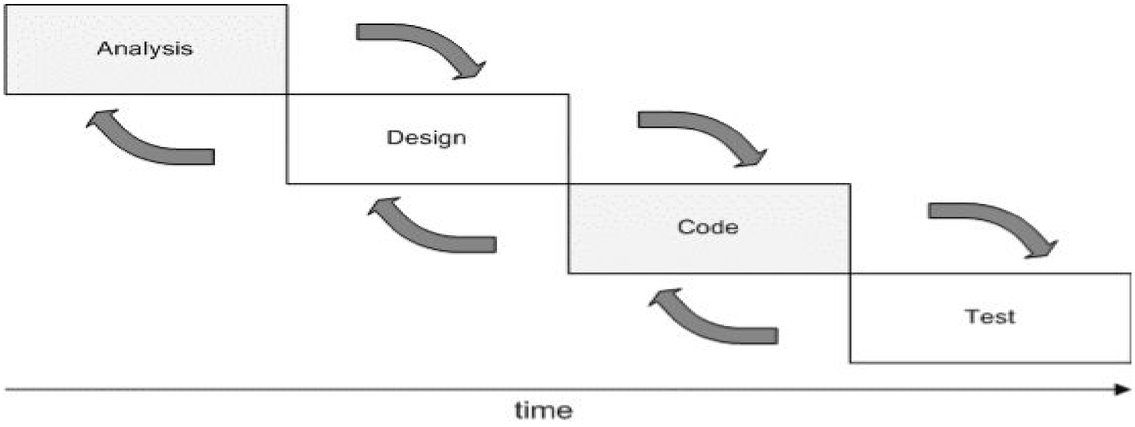


Figure 3. Waterfall Model (From: [7])

2. The Prototyping Model

“The prototyping model is based on developing one or more prototypes to clarify software requirements. There are two types of prototypes: throwaway and evolutionary prototyping [7].” In prototyping, most prominent parts are designed first. It is good to use where requirements are changing rapidly, and the problem domain is not clear [6].

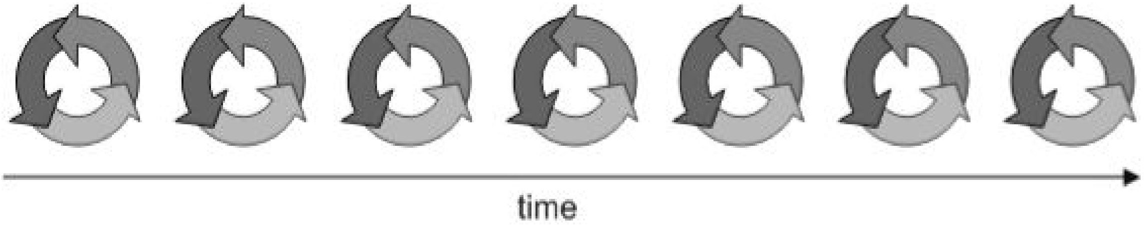


Figure 4. Prototyping Model (From: [7])

3. Rapid Application Development Model

Rapid Application Development (RAD) model is based on developing an application within a short time. The project team develops the application in design meetings. The built application is evaluated by using prototypes. The team uses reusable software components in building the design [7]. Rapid Application Development model is a like a high-speed waterfall design. RAD develops the product component-by-component [8]. RAD uses extensive user input in software design and is good for systems where extensive user contribution is available [6].

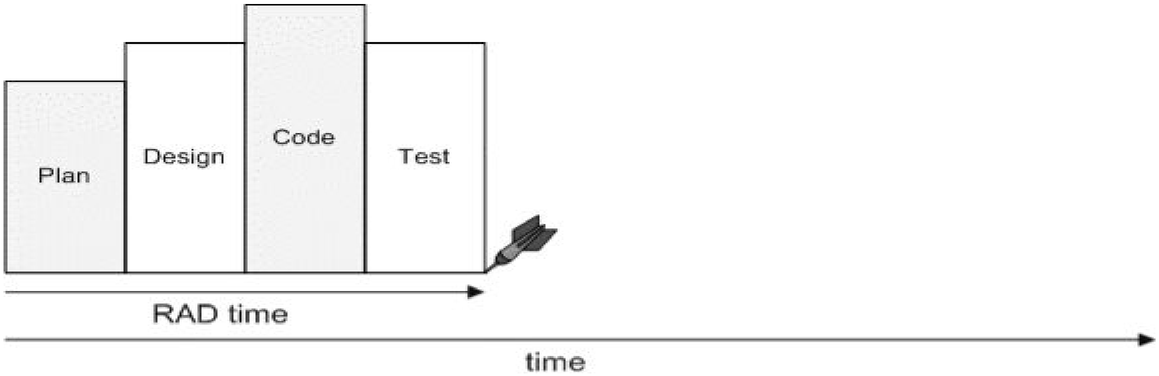


Figure 5. RAD Model (From: [7])

4. Spiral Model

“The Spiral model involves a series of increments in a cyclic process, works better with internal software development than contract software development, due to less flexible contract mechanisms [7].” Spiral model is like “a series of mini projects [6].” Spiral models are good for projects where requirements are not well known [8].

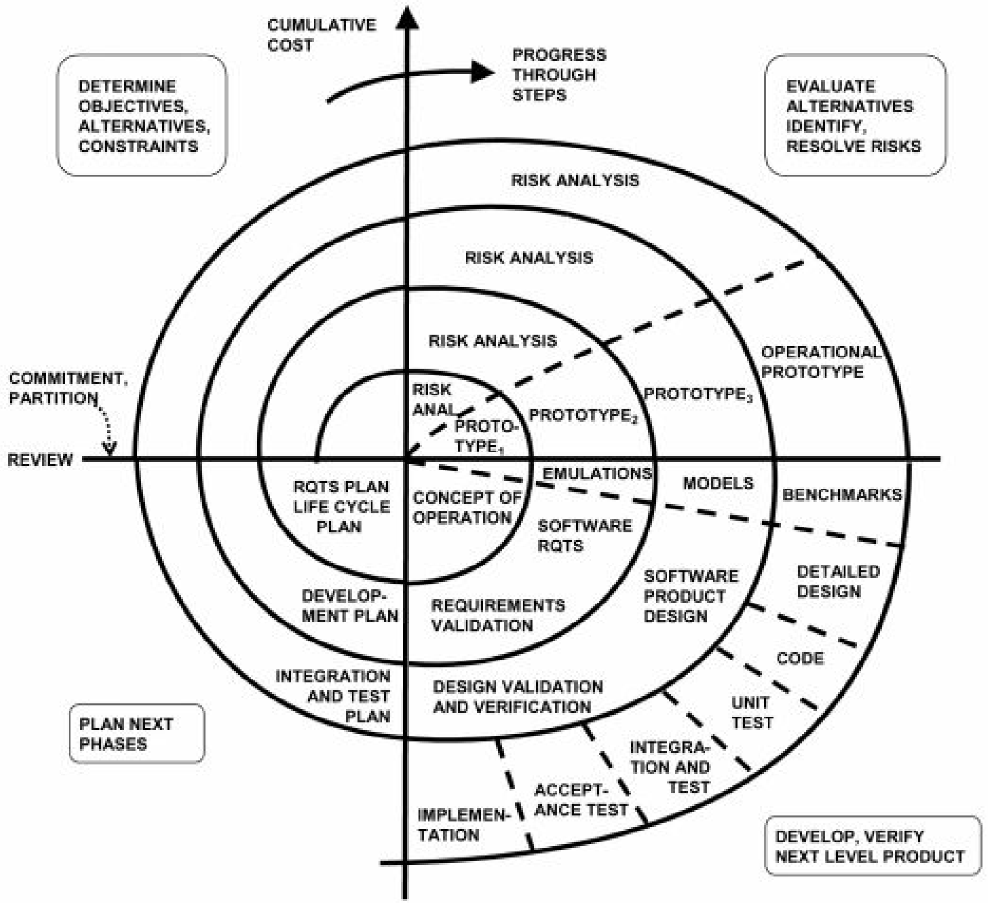


Figure 6. Spiral Model (From: [6])

5. Rational Unified Process

The Rational Unified Process (RUP) comprises some of the best practices in software development. These best practices include iterative development, requirements management, component-based development, visual modeling with UML, quality management, and change management [7]. RAD has 4 phases, which are inception, elaboration, construction and transition [6]. Each of the RUP phases comprises a series of activities. These activities are divided into “core” and “supporting” workflows. Core workflows include business modeling, requirement analysis and design, implementation, test, and deployment. Supporting workflows include project management, configuration and change management, and environment [9].

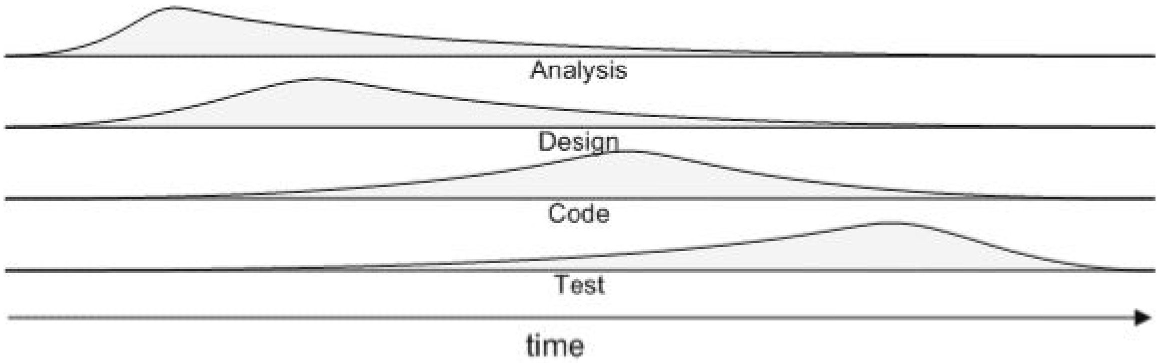


Figure 7. Rational Unified Process (From: [7])

6. Extreme Programming

“Extreme Programming (XP) programmers perform small pieces of planning, design, coding, and testing at times throughout the development lifecycle [7].” Extreme programming practices include planning game, small releases, metaphor, simple design, testing, refactoring, pair programming, collective ownership, continuous integration, 40-hour week, on-site customer and coding standard [10]. Extreme programming requires good and experienced developers [6].

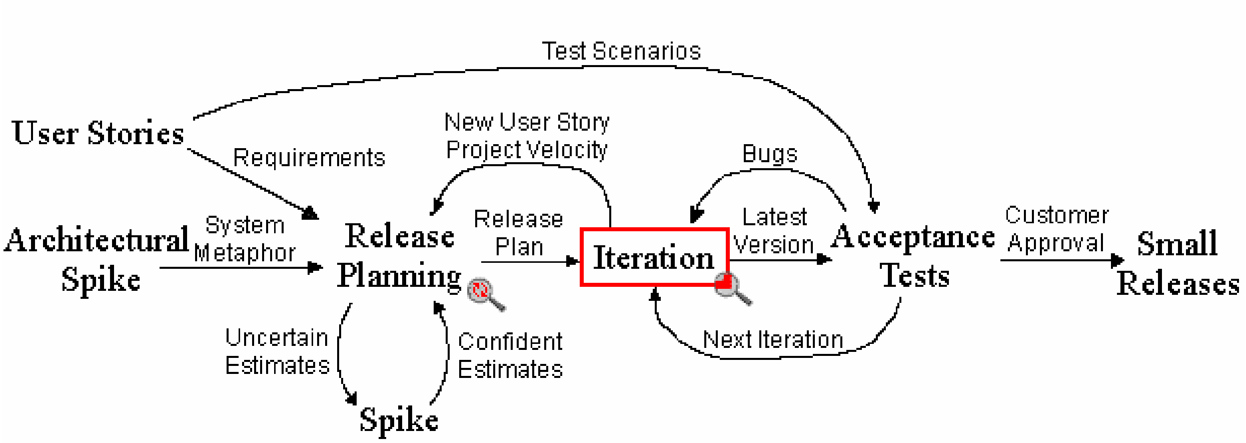


Figure 8. Extreme Programming (From: [6])

Here spike in Figure 8 refers to a set of requirements, or other things for the next phase.

There are other methodologies used other than cited above: “Dynamic Systems Development Method in Europe; Feature-Driven Development in Australia; and Extreme Programming, Crystal, Adaptive Software Development (ADP), and SCRUM in the US [7].” “Manifesto for Agile Development” written by the practitioners of these methodologies can be found in their website www.agilemanifesto.org [7].

III. FEATURES AND REQUIREMENTS FOR A PROJECT MANAGEMENT MODELING LANGUAGE

A. WHY THERE IS A NEED FOR MODELING THE PROJECT MANAGEMENT?

Before answering why we need to model project management, it will be appropriate to describe what is meant by a visual tool. A visual tool is a model that depicts a project in graphical representations. These representations may depict a part or phase of a project, and the entire project in a few pages of graphics.

Human beings sense the world with five different organs. The most important one of these organs is the eye. Humans' perception of the world is mostly achieved with the eyes. A visual Software Project Management tool will be helpful for each individual in a project to understand what is going on more easily then other techniques. With the help of a visual tool, individuals can easily follow the phases of a project, their individual roles, current phases, and future phases.

Project management is complex and comprises various aspects. These aspects include cost, schedule, quality, people, procurement, risks, etc. The project manager's job is to plan, control, execute and close the project by successfully managing all those aspects. Those aspects are dynamic, not static. They change over time and the manager has to deal with those diversions from the original plan. Visual modeling tools are aids that ease and help project manager in controlling various aspects of the project. "A picture is worth a thousand words [11]." A picture is easy to follow, helps in communicating the project, and easy to build.

Stakeholder buy-in is crucial to project success. To provide stakeholder buy-in, the manager has to communicate and coordinate the project effectively. The project manager has to manage, convince, and motivate people. His job is to sell the project to stakeholders including the project team. Stakeholders comprise any individual related to the project including owners, managers, end-users, developers, project team, etc. The project manager is responsible for convincing and motivating stakeholders to participate and give their best for the project. He/she also has to convince the owners that the project will bring value to their organization and it will be completed within a reasonable time, at a reasonable cost, and with acceptable desired quality. If the manager fails to obtain stakeholder buy-in, unpredictable problems occur resulting in delays and resulting in the cancellation of the project. A visual tool that includes activities, inputs, and outputs of a project and the roles of the stakeholders in those phases can help in stakeholder buy-in. If stakeholders can easily follow what is going on, and what their individual roles are, stakeholder buy-in may be easier.

Communication is a significant challenge area for a project manager. When the manager fails to communicate the project, it is possible to face resistance from the organization, executive managers, and even from the project team. It is important that stakeholders have a general understanding of the project and its profit to the organization and to the individual. Written documents can be long, complex and hard to understand. They reflect the terminology of the writer, but most of the time stakeholders in a project have different backgrounds and the writer’s terminology may be unfamiliar. Even if the terminology is familiar, it will take time to read and figure out the overall picture of the project. People are not willing to read long written documents, and in project meetings, it is not easy to focus their attention on what is going on. With the help of a visual tool, depicting the entire project in a few schemas and depicting responsibilities and relations of the individuals to those phases can be helpful in effective communication of the project plan and progress of the project to stakeholders.

B. WHAT ARE THE FEATURES OF EXISTING PROJECT MANAGEMENT TOOLS?

There is no existing project management tool that address all aspects of a project. Existing tools addresses certain aspects. Some address schedules and activities. Some address relationships between activities. Some address technical aspects of the project. However, when it comes to management, there is no single tool that addresses activities, entities, stakeholders, cost, schedule, quality, etc, together. Here in this section, we will describe existing tools widely used in project management and their features.

1. Work Breakdown Structure

A work breakdown structure (WBS) is a graphical or textual tool used to depict the hierarchical decomposition of project into phases, activities and tasks. "A WBS takes the project and divides it into smaller pieces [12]." WBS decomposes a project into tasks, subtasks, processes. WBS is about identifying what needs to be done [6]. Hierarchical decomposition of the project eases planning, control and execution of the development since it is easier to manage small parts rather than entire project. The project management team develops the WBS with the project team based on determined (hopefully agreed with stakeholders) scope and objectives of the project.

WBS is a basis for many things including cost estimating, scheduling, risk analysis, organization structure, project control, and planning. WBS allows the project team to determine budget and schedule of the project more precisely [2]. Making estimates based on small and more manageable components can give better results rather than estimating based on the entire project.

Work breakdown structures are not limited to tasks. A WBS can be developed based on entities, activities, products or deliverables of the project. A product WBS decomposes products into sub-products. For large projects, building a product breakdown structure makes the construction of a WBS much easier. Since Work Break Down Structures are graphical, simple and easy to read, they are valuable aids for project managers in communicating the projects to the stakeholders [12].

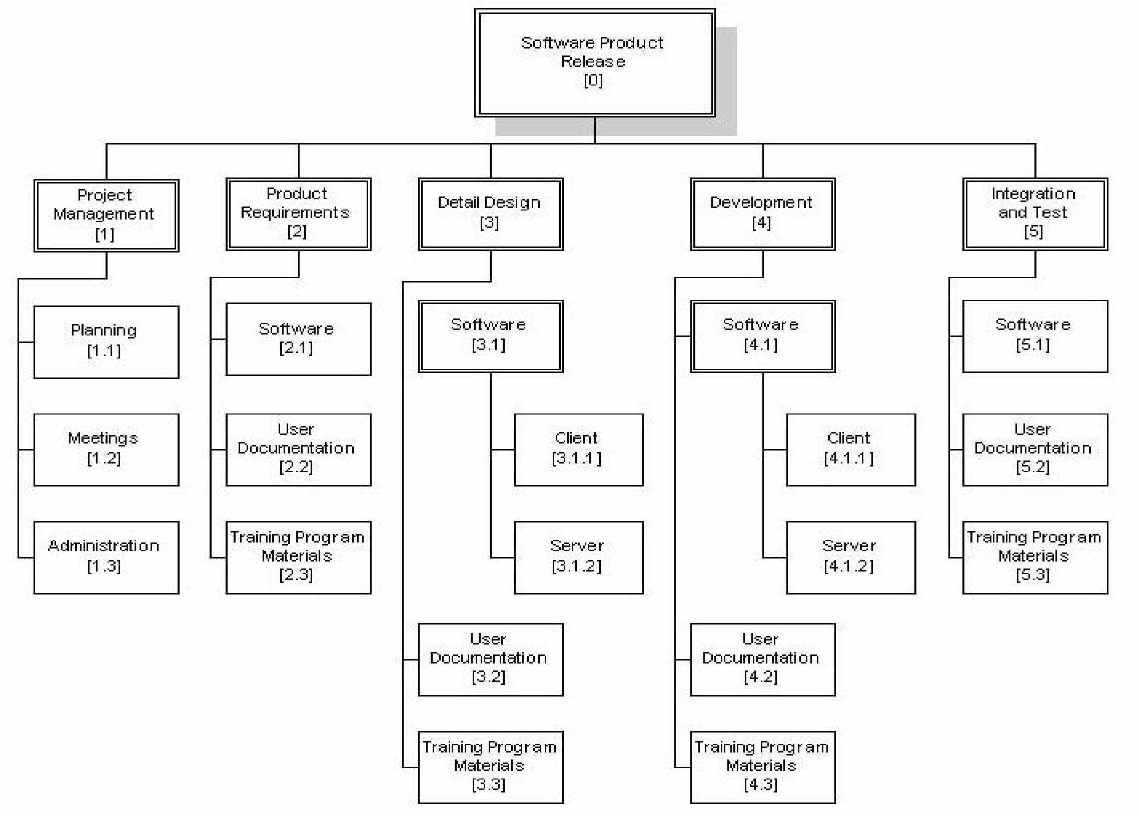


Figure 9. A Sample WBS (From: www.cit.cornell.edu)

A WBS helps the project manager in planning the project, allocating resources, assigning people to tasks and following activities that need to be done in order to complete the project. However, WBSs exclude inputs and outputs of activities or tasks and their relations to other tasks. A WBS also lacks the dependencies between activities. Stakeholders of the project and their relationships and responsibilities to any activity or task in the project cannot be determined from a WBS, but the WBS can be used as a basis for determining the responsibilities and assigning people. WBSs are quite helpful; however, they address only certain aspects of the project.

2. Gantt Charts

Gantt charts graphically depict the sequence of activities in a project on a timeline. They show activities, their start and end dates. Gantt Charts are helpful in controlling the project schedule. Gantt charts consist of a horizontal and a vertical axis. On the horizontal axis lays a calendar. Vertical axis lists the tasks of the project. Each task is represented by a rectangle. The left side of the rectangle positioned over the start date and right side over the end date [12].

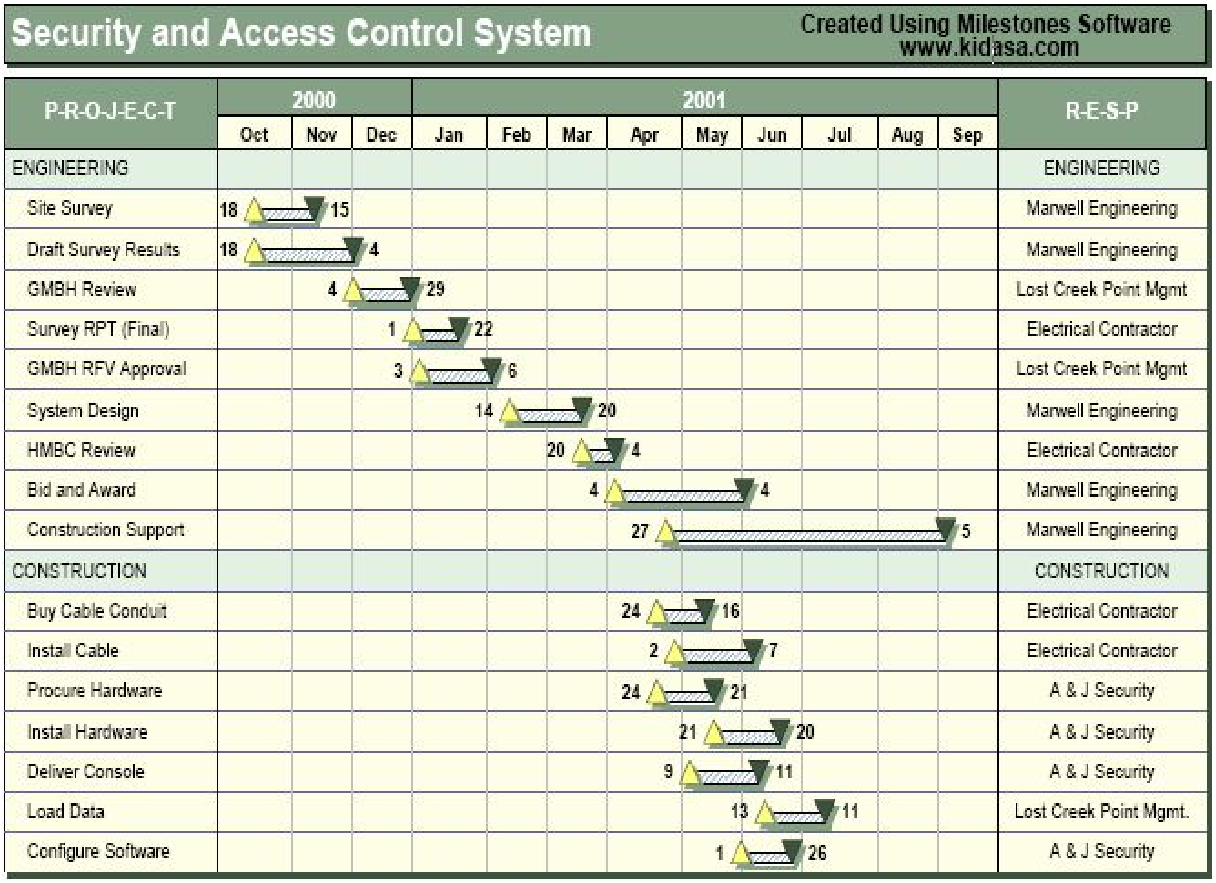


Figure 10. Example Gantt Chart (From: www.kidasa.com)

They do not show inputs and outputs of the activities. Deliverables from previous activities, requirements and inputs to following activities and relationships between those are excluded. They may show dependencies only between activities. They also exclude stakeholders and their roles in activities. Modifications can be made to add more features to Gantt Charts, but means diversion from the original purpose, which is simplifying and easing the understanding of project management.

A Gantt chart is useful for planning purposes but it has a limited scope. It is a reality that a project will not follow the planned schedule most of the time. Timelines of the Gantt chart will require continuous updates. The benefit of a Gantt chart will be limited to showing sequence of events.

When several parallel activities are present in the project, Gantt charts can grow very big which makes them hard to follow.

3. Pert Charts & CPM

PERT stands for Program Evaluation and Review Technique. Newell and Grashina define PERT as a statistical approach to project schedules. PERT is used for project schedule estimations where task durations are uncertain [12].

Activities depicted on PERT charts depend on three estimations. These are optimistic, pessimistic and the most likely durations. These three values give expected durations and standard deviations of the activities. Project schedule estimation is calculated and determined based on expected durations and standard deviations [12].

PERT charts show the sequence of activities, the tasks that must be performed in parallel, critical path of activities that must be completed to meet the deadline, and dependencies between those activities. They are especially useful when there are parallel activities or subprojects that must be completed simultaneously. They give earliest and latest start and end dates for each activity [12].

Critical Path Method (CPM) determines the activities that should be completed on planned schedule in order not to delay the completion of the entire project. CPM shows on which activities the project manager should focus. CPM can be used in parallel with PERT charts. The Critical Path is shown with thicker lines to depict activities connected with those lines are critical to meet the schedule. There may be more than one critical path [8].

CPM is based on one estimate, and PERT is based on three estimates. PERT is a probabilistic method and CPM is deterministic [6].

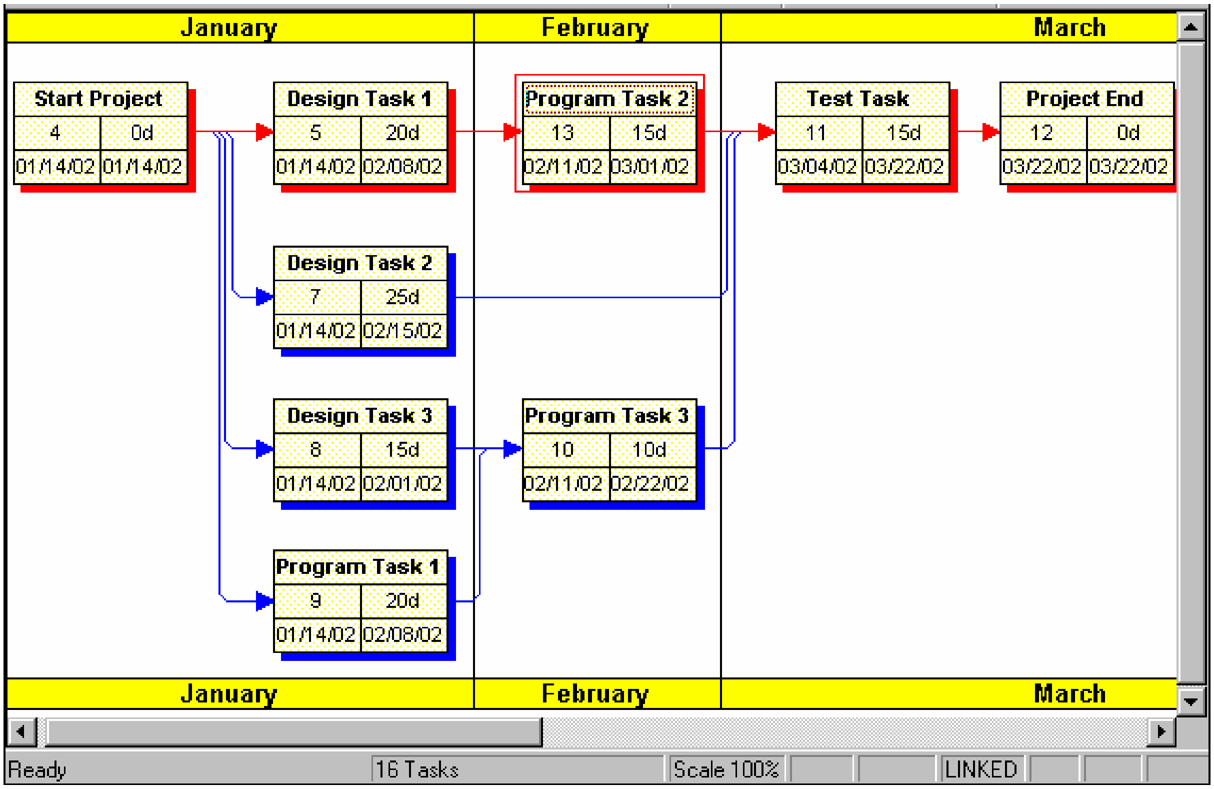


Figure 11. A Sample PERT Chart (From: www.criticaltools.com)

PERT charts and CPM mainly focuses on schedule aspect of the project. They are based on activities and dependencies between those activities. However, they lack the stakeholders of the project and their relations and responsibilities to any activity or activities. They do not show the inputs or specifications that development team must address or use as a reference. They also lack the outputs or deliverables of any activity that will be an input to following activities.

4. Unified Modeling Language (UML)

Unified Modeling Language (UML) is a visual language for analyzing and designing object-oriented systems. UML is used to visualize, and document the models of the software systems and organizations using such systems [13].

UML is a technical modeling tool that depicts the technical aspect of a project in a visual model. It helps to follow the data and activity flows, relationships between objects and activities, etc. UML uses separate diagrams for separate design specifications related to physical design specifications of the system.

The project team develops the UML diagrams of a project. These diagrams include use-case diagrams, class diagrams, sequence diagrams, activity diagrams, state-chart diagrams, and implementation diagrams [13]. Developers use those diagrams as references to physical development phases of the software project.

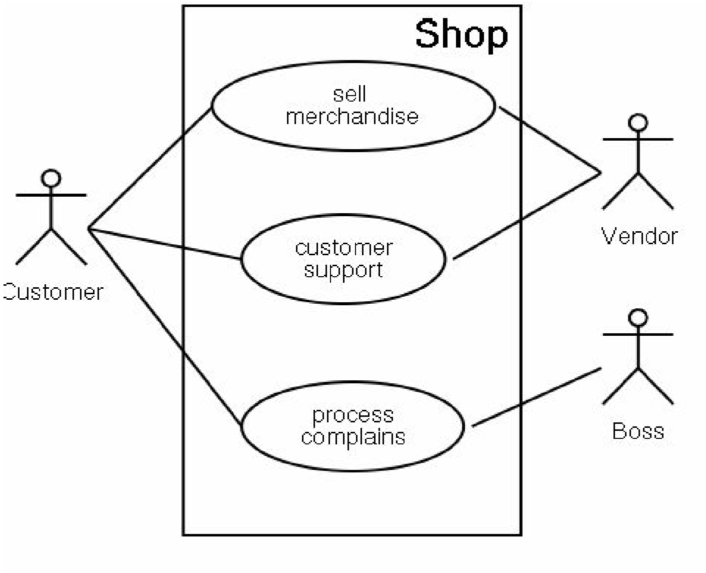


Figure 12. A Sample Use-Case Diagram (From: www.pms.ifi.lmu.de)

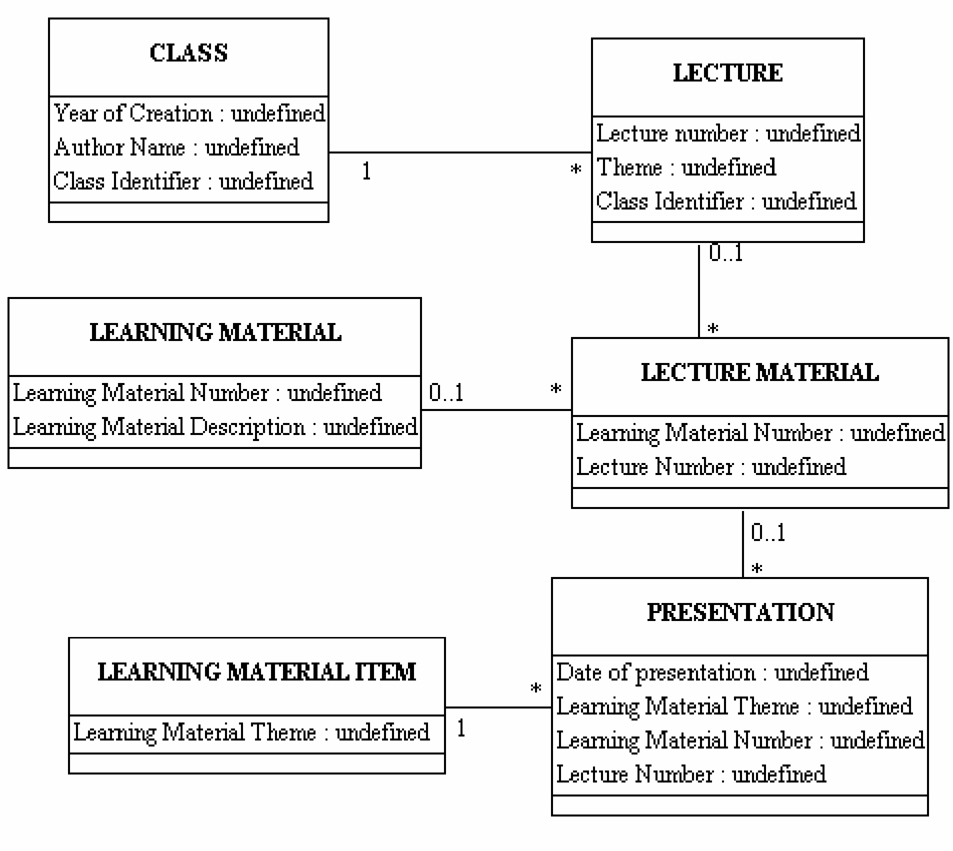

CLASS
Year of Creation : undefined
Author Name : undefined
Class Identifier : undefined
1
*
LECTURE
Lecture number : undefined
Theme : undefined
Class Identifier : undefined
0..1
*
LEARNING MATERIAL
Learning Material Number : undefined
Learning Material Description : undefined
0..1
*
LECTURE MATERIAL
Learning Material Number : undefined
Lecture Number : undefined
0..1
*
PRESENTATION
Date of presentation : undefined
Learning Material Theme : undefined
Learning Material Number : undefined
Lecture Number : undefined
LEARNING MATERIAL ITEM
Learning Material Theme : undefined
*
1

Figure 13. A UML Class Diagram (From: www2.sims.berkeley.edu)

UML diagrams are related to physical design specification and appropriate for physical design phase of the project. However, they do not show management aspects, activities, entities and stakeholders related to development lifecycle, and relations between those. It is possible to modify the standard notations used in UML and model the development phases and relations. However, this would be a new subject to work on.

UML diagrams depict different aspects of the project in different diagrams. These diagrams may get complex and hard to follow for non-technical personnel.

5. SysML

SysML is an object oriented modeling tool similar to UML. “The SysML extends and customizes UML 2 to support systems engineering activities in engineering of complex systems [14].”

SysML comprises and extends UML 2 diagrams. SysML enhances composite structure and activity diagrams and brings two new diagrams, which are requirements and paramedic diagrams. SysML is a modeling language for systems engineering. Osmundson and Hunyh state that SysML is effective in specifying requirements, system structure, functional behavior, and allocations during specification and design phases of systems engineering [14].

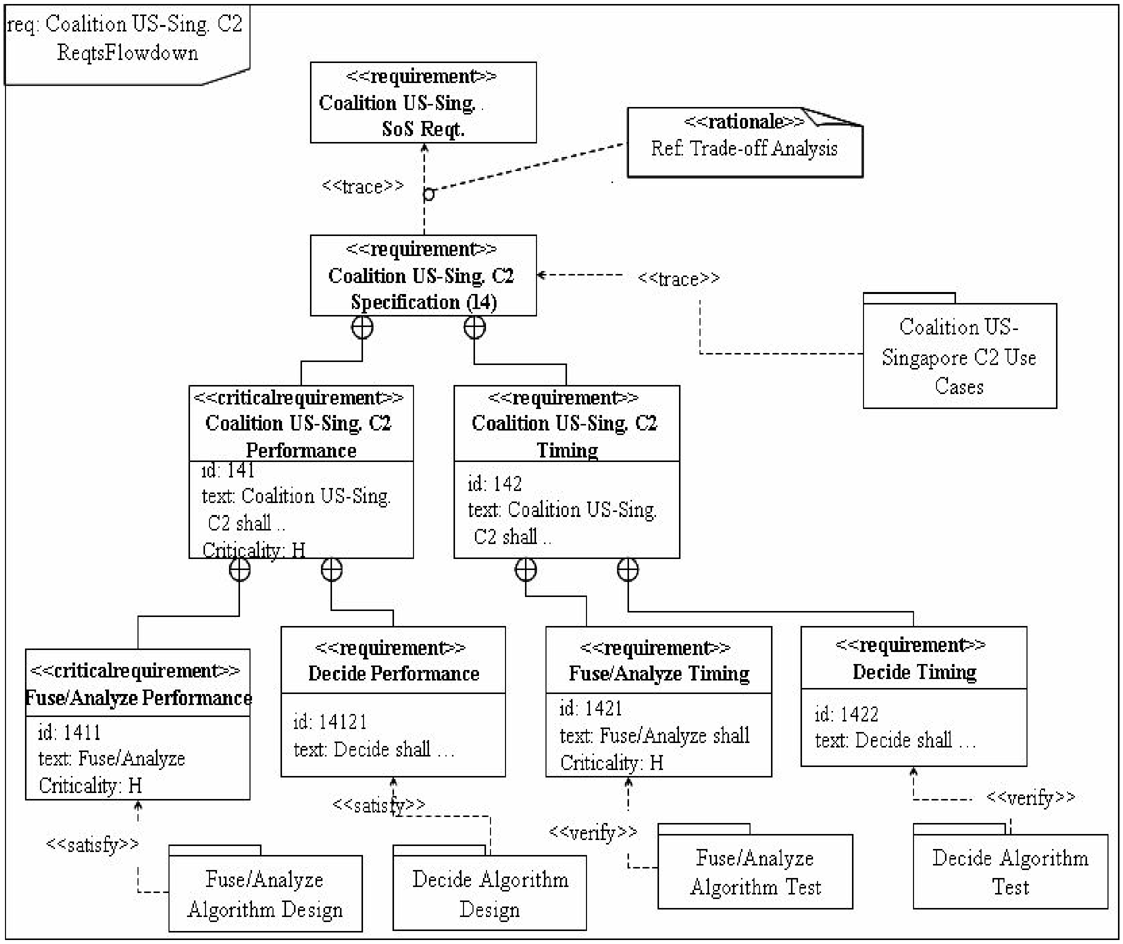


Figure 14. A SysML Requirements Diagram (From: [14])

As it can be seen from descriptions like UML, SysML is also related to the physical design specifications of the proposed system. They are helpful in management purposes but they are about the technical aspects of the system, not management. They also exclude stakeholders, their relations and responsibilities in development cycle.

6. Available COTS Software

There are a number of available project management software in the market providing project managers a set of tools. These tools are generic and do not address a specific management discipline. In general, they provide platforms for planning, controlling and executing projects. Some of the available software are discussed below.

a. Microsoft Project

Microsoft project provides a wide set of tools to project managers. These tools help project managers to build schedules, allocate resources and manage the budget of the project. The software comprises well-known project management tools. The software provides generic platforms for the project managers to build PERT, GANT, WBS, CPM and organization charts. All of these charts are related to each other and can be converted from one to another. Dependencies between the activities can be identified from the charts. Schedule and cost can be assigned to any activity, and this provides a baseline for cost and schedule analysis [15].

The software provides valuable tools to project managers. However, the project manager needs to navigate between various tools to see the complete picture of the project. This brings complexity to project management. Inputs and deliverables of the activities are not addressed in the software.

b. OpenProj

OpenProj is a free open source alternative for MS Project. OpenProj provide tools for a project manager to plan and control a project. Gantt Charts, PERT charts, WBS, and Earned Value costing are some features that OpenProj provides. The project manager can plan and follow project schedule, control project budget and assign tasks with the tool [16].

OpenProj sets a separate interface for each popular project management tool. The software is for the use of the project manager as an aid in his or her job. The software puts existing software management tools into a package. Inputs and outputs of the projects are not covered in the software. The software is not intended for the use of stakeholders other than the project team.

c. Attask

Attask is a web-based project management software. Like MS project, provides templates and tools for building GANTT, PERT, WBS, organization charts and CPM analysis. All these features are built in separate interfaces, but can be interacted. Attask allows you to assign team members to tasks and lets you define dependencies [17].

Attask can interoperate in collaboration with MS project. The software provides all of the project management features that MS Project comprises. In addition to that, the software includes resource and risk management features [18].

The software targets the project team, and requires an experienced team to fill out the interfaces of the project management tools. However, stakeholders, inputs and deliverables are not included in the software.

d. Project.net

Project.net is an open source web-based project management software. Project.net comprises features to plan and control budget and schedule of the project. The program provides tools to prepare graphics depicting what is going on with the project. You can assign individuals to tasks and define their responsibilities. Project.net is mostly report based rather than visual [19].

e. Daptiv

Daptiv is a web-based project management software. The program provides basic platforms for budget, schedule and task control. Daptiv also comprises resource management, collaboration and communication management and risk management features [18].

Daptiv is a comprehensive software application including platforms for most of the software project management related areas. However, these platforms are mostly textual rather than graphical.

f. Celoxis

Celoxis is a browser-based project management software. Program features involve project planning, WBS, Gantt charts, estimation, cost management, budget management, critical path and earned value analysis. Celoxis is focused on task management [18].

Some other available project management software are e-studio, Copper, Project Kick-Start, Smooth projects, Milestones and Minuteman, Open Workbench, Bugzilla, Tracker Suit, Basecamp, etc.

Project management software provides valuable platforms to project managers. Almost all of the software combines existing and proven software project management tools discussed above. However, users have to navigate between these platforms to see the complete picture of the project. Some of them cover stakeholders and let you assign tasks to individuals. Inputs and deliverables of the project are not covered in these software packages.

C. GUIDANCE CRITERIA FOR DEVELOPING SOFTWARE PROJECT MANAGEMENT TOOLS

Until now, we discussed the need for a new tool, existing tools and their features, and what are absent in existing tools. In this section, we discuss the required features for a new modeling tool. The goal of this section is to form a framework for a software project management modeling aid.

1. The Modeling Tool Should Be Simple

We discussed before that complexity of project management with diverse aspects causes problems for a project manager in controlling and execution phases of the development cycle. The new tool should depict adequate information that any individual related to the project can understand quickly. It should be able to visualize information that is documented in tens of pages. Being simple does not mean excluding some aspects. Besides being simple, the model should be comprehensive enough to extract required data by anybody participating in the project.

2. The Modeling Tool Includes All Aspects of the Software Projects

PMBOOK identifies related areas to project management that a project manager should address as integration management, scope management, time management, cost management, quality management, human resource management, communications management, risk management and procurement management [1]. In essence, project manager has to deal with scope, budget, cost, quality and people [2]. Project managers' responsibility is to complete the project within defined scope, expected quality, supplied budget and estimated schedule while dealing with stakeholders and the project team. New modeling tools should be able to address these areas. A modeling tool should include not only stakeholders but also their relationship to any activity or entity.

3. The Modeling Tool Depicts the Work Breakdown Structure of the Project

All of the tasks and subtasks comprising the project should be derivable from the model. When an individual examines the model, he should be able to see tasks, subtasks, performers of these tasks, and relationships between those tasks. He or she should be able to see his/her responsibilities and co-workers, supervisors, etc. In essence, model should also depict the organization chart in conjunction with WBS.

4. The Modeling Tool Depicts the Inputs and Outputs of Any Activity

Inputs are entities or specifications that are required to complete a task. Inputs include requirements that are determined by project team by interviewing with stakeholders and end-users. The modeling tool should show requirements for any activity. When builders start developing a part of the project, they should be able to see the requirements for the outputs when they analyze the model. Deliverables from previous activities are also inputs to following activities. Those deliverables are also outputs of the activities. The final deliverable is the completed project.

5. The Modeling Tool Permits Formal Analysis

Glatstein defines formal analysis as a technique used for organizing visual information. It is a strategy used to translate visual information into written words [20]. Any individual examining the model should be able to understand the project, its specifications, phases, activities, entities, etc. He should be able to define, describe, analyze and interpret the project.

6. The Modeling Tool Should Support Existing and Future Project Management Methodologies

Popular project management methodologies are described above in this chapter. They are proven methodologies that help in planning, controlling and executing the development phases. The model should reflect the specifications of available and proved project management theories. The model should use existing management models as a basis. Project managers should be able to depict any process, entity, activity in the existing models with the tool.

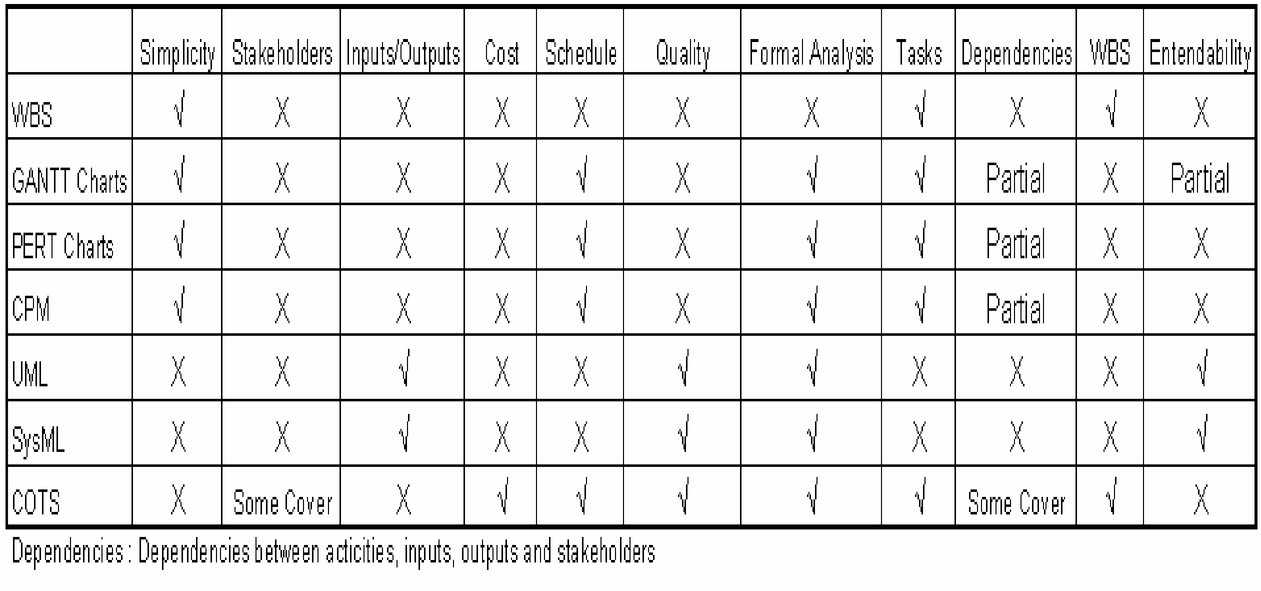
7. The Modeling Tool Should Be Extendible

The modeling tool should be open to new enhancements and changes in the project. It is obvious that any project will have diversions from the original plan. Requirements and scope of the project may change as the project goes on. Project may grow or shrink in size. New activities should be added or some activities should be omitted. New stakeholders may have to be added, or some of them may leave the project. The model should be dynamic to reflect any change to the project.

8. The Modeling Tool Should Show Stakeholders and Their Responsibilities and Relations to Any Activity

Stakeholders are crucial to project success. Stakeholders comprise anybody involved in the project from the project team to owners, managers, users, etc. Modeling language should be able to depict stakeholders participating in any phase, activity or task of the project. The model should also depict the roles and relations of these stakeholders to activities.

Following table demonstrates the models discussed above and their features based on management:



	Simplicity	Stakeholders	Inputs/Outputs	Cost	Schedule	Quality	Formal Analysis	Tasks	Dependencies	WBS	Entendability
WBS	√	X	X	X	X	X	X	√	X	√	X
GANTT Charts	√	X	X	X	√	X	√	√	Partial	X	Partial
PERT Charts	√	X	X	X	√	X	√	√	Partial	X	X
CPM	√	X	X	X	√	X	√	√	Partial	X	X
UML	X	X	√	X	X	√	√	X	X	X	√
SysML	X	X	√	X	X	√	√	X	X	X	√
COTS	X	Some Cover	X	√	√	√	√	√	Some Cover	√	X

Dependencies : Dependencies between acticities, inputs, outputs and stakeholders

Table 1. Features of Existing Tools Based on Desired Criteria

THIS PAGE INTENTIONALLY LEFT BLANK

IV. SOFTWARE PROJECT MANAGEMENT THEORY

A. SOFTWARE PROJECT MANAGEMENT THEORY AND MODELING LANGUAGE

This chapter is based on the article “A Novel Project Management Modeling Tool” [21]. The theory aims to model management of software projects to help software project manager in planning, executing, controlling and closing the project. Its goal is to provide a formal tool to analyze projects and project management. The benefits of the theory include:

- Simplify project management complexities using basic concepts.
- The theory has explanatory power of any type of projects. It is not customized for any specific type.
- It provides a formal modeling tool. The tool is also supported by a graphical representation to ease the understanding and usage.
- The modeling tool is extendable.
- The theory enables us to formally define and analyze projects. Using the modeling tool, it is possible to conduct various analysis and investigate project management best practices within projects.
- Both static and dynamic analysis of projects can be conducted using the theory.
- By Using the modeling tool, it is possible to create project histories and databases to enable further research on project management.

B. PROJECT MANAGEMENT THEORY BASICS

The definition of a project and project management within the theory is as follows:

A project is an output of a project management function. The inputs for this function are a limited number of activities and entities related to any part of the project. An activity is a named process, function, or task that occurs over time. An entity is something that has a distinct, separate existence, though it need not be a material existence.

In theory, project management is viewed as a mathematical function. The formula of project management is given in Relation 4.1. Equation Chapter 4 Section 1

$$P = PM\left\{a_1(), a_2(), a_3(), \ldots, a_m(), e_1, e_2, e_3, \ldots, e_n\right\} \tag{4.1}$$

In Relation 4.1, *P* denotes the project, and *PM* is the project management function that outputs the project. The inputs of the project management function are activities, denoted by $a(\ )$, and entities represented by e.

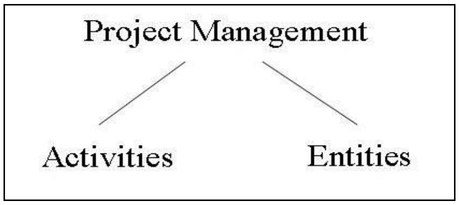


Figure 15. Activities and Entities.

Two important concepts lies in the heart of the theory as depicted in Figure 15: Activities and entities. Examples of activities are requirements analysis, testing, stakeholder analysis, prototyping, staff meetings, code reviews etc. Examples of entities are project manager, staff, teamwork, test cases, leadership, requirements, documentation etc. Using these two important concepts, it is possible to define and explain any project with a management emphasis.

C. RELATIONS AND THE MODELING LANGUAGE

To develop the modeling language, we defined relations between these two important concepts that make up the theory. Some of these relations help to define projects while some are useful for analysis purposes. The modeling tool is expandable with the introduction of new relations as long as they do not contradict with previous relations.

1. Create

Within a project, an entity can be created resulting from an activity. Therefore, it is possible to define a create relation between an entity and an activity as in Relation 4.2.

$$e_y = a_x(\ ) \tag{4.2}$$

For example, staff is an entity that can be created as the result of a hiring activity as explained in Figure 16. The create relation is one of the basic relations of the modeling language.

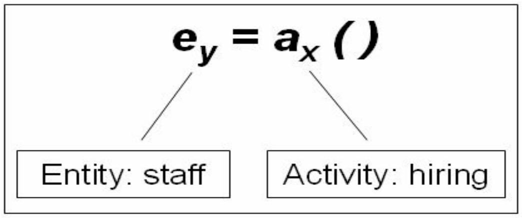


Figure 16. A Create Example

2. Transform

The relation transform is also another basic relation defined with the modeling language. An entity can be transformed to another entity as a result of an activity. Relation 4.3 presents the formulation.

$$e_y = a_x(e_1, e_2, e_3, \ldots\ldots, e_n) \tag{4.3}$$

For example, the entity project specification, can be transformed to the entity project design documentation, as the result of the design activity. The example is shown in Figure 17.

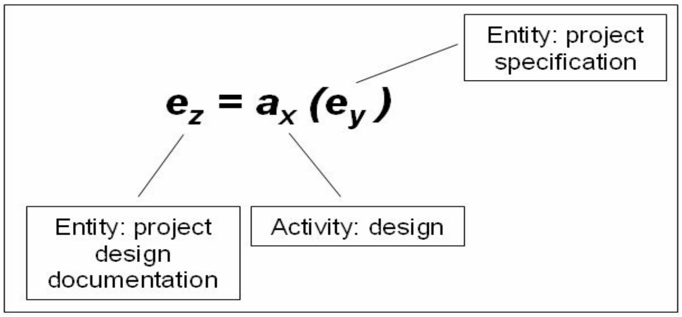

Entity: project specification
$e_z = a_x (e_y)$
Entity: project design documentation
Activity: design

Figure 17. A Transform Example

At the highest level, a transform relation can also explain the project management function. A project is an entity while project management is the activity. The input for this activity is the entity business need. Every project starts with a business need. If there is no need for the project, it means the project does not have a use. At this highest level, we have a basic model of a project. This basic model is shown in Figure 18.

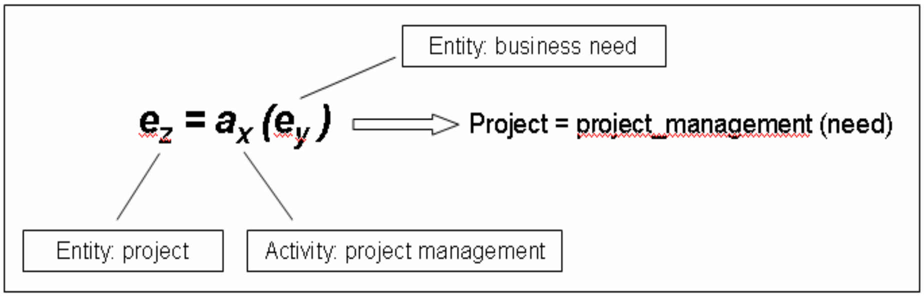

Entity: business need
$e_z = a_x (e_y)$
Project = project_management (need)
Entity: project
Activity: project management

Figure 18. Basic Model of Project Management

3. Delete

An activity or an entity can be deleted during a project. This represents a delete relation. For example, when a project manager quits the job or gets fired, such an incident is modeled with this relation. The proposed modeling language also supports dynamic analysis. The delete relation helps to model such dynamic behaviors. Another example is the removal of a feature or a component from a software product. Actually, the delete relation can be thought of as a special activity within a project. This special activity is represented with DELETE. Relation 4.4 shows the formulation.

$$DELETE(e_n) = \varnothing = \; ; \quad DELETE(a_n()) \quad \varnothing \qquad (4.4)$$

Also note that specialized terms are denoted by capital letters such as P for project, PM for project management and DELETE for delete.

4. Divide

An activity or an entity can be divided into smaller activities or entities. This relation is denoted by the word DIVIDE. For example, the activity testing can be divided into smaller activities such as subcomponent testing and integration testing activities. The formula of a divide relation is shown in relation 4.5.

$$\begin{gathered} DIVIDE(a_m()) = \{a_1(), a_2(), a_3(), ..., a_n()\} \\ DIVIDE(e_m) = \{e_1, e_2, e_3, ..., e_n\} \end{gathered} \qquad (4.5)$$

5. Aggregate

The relation aggregate represents the relation of combining smaller activities or entities to an activity or to an entity. The representation for the relation aggregate is the word AGGREGATE. The relation is given is relation 4.6.

$$\begin{gathered} a_m() = AGGREGATE(a_1(), a_2(), a_3(), ..., a_n()) \\ \mathrm{e}_m = AGGREGATE(e_1, e_2, e_3, ..., e_n) \end{gathered} \qquad (4.6)$$

For example, the entities schedule, staff requirement, and cost estimation can be aggregated to the entity project plan. Figure 19 presents the example.

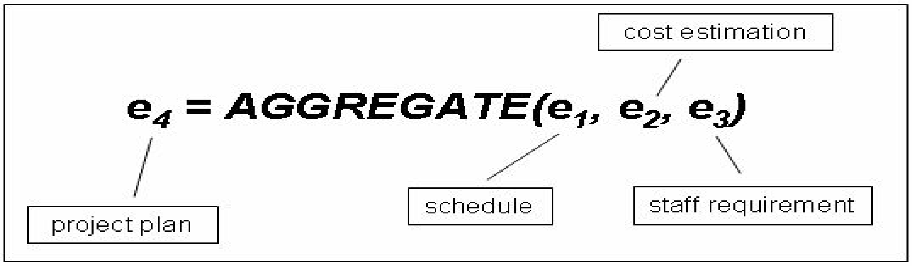


Figure 19. An Aggregate Example

Examples:

$$a_n() = AGG\mathrm{R}EGATE(a_1(), a_2(), a_3())$$
$$e_4 = AGGREGATE(e_1, a_2(), a_3())$$

6. Next

An important aspect of projects is the necessary arrangement of activities and entities. The relation next helps us to define such arrangements within projects. Activities and entities can be followed by another activity or entity. This is modeled by the relation next and it is denoted with a right arrow sign (→). For example, in a software project, it is essential that the coding activity be followed by a testing activity. Relation 4.7 shows the relation.

$$a_1() \rightarrow a_2() \quad ; \quad \mathrm{e}_1 \rightarrow e_2 \tag{4.7}$$

7. Previous

The relation previous is similar to the relation next. When activities and entities are preceded by another activity or a relation, the previous relation is used. This relation is denoted by a left arrow sign (←). An example of this relation is that the entity design must be preceded by a requirements analysis entity. The formula is in Relation 4.8.

$$a_2() \leftarrow a_1() \;\; ; \;\; e_2 \leftarrow e_1 \tag{4.8}$$

Figure 20 shows a next and previous relation example.

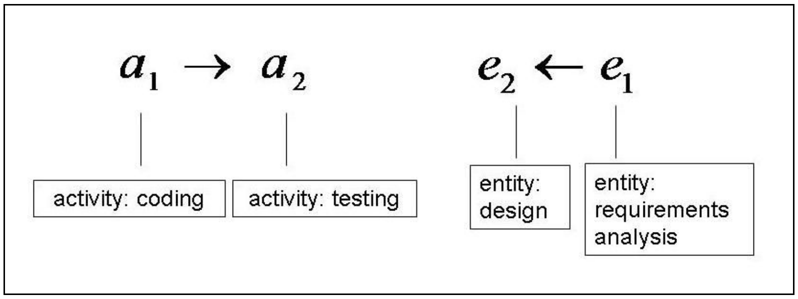


Figure 20. An example of the relation next and previous

8. Require

When an activity or an entity is required in order to exist by other activities or entities, require relation can be used. For example, in order to conduct the testing activity within a software project, it is required to have a requirements document and a coding activity. This relation is denoted by REQUIRE. The relation formulation is presented in relation 4.9.

$$REQUIRE(a_m()) = \{a_1(), a_2(), a_3(), ..., a_n()\}$$
$$\text{Activity } a_m() \text{ requires activities } a_1, a_2, ..., a_n. \qquad (4.9)$$

9. Exist

This relation is especially important to conduct analysis on the modeled project. Using requires relation it is possible to check for faults in a project. In addition, it is possible to search for formulized best practices in a specific project model.

To verify whether an entity or an activity exists within a project model for analysis purposes, the relation exists is given in relation 4.10. This relation is represented by EXIST.

$$EXIST(a_n()) = True / False \quad ; \quad EXIST(e_n) = True / False \qquad (4.10)$$

10. Decision

This relation determines any situation where a decision is required at any part of the project.

$$DECISION(e_1, e_2, ..., e_m) = \begin{Bmatrix} \{decision_1, a_1()\}, \\ \{decision_2, a_2()\}, \\ \{decision_3, a_3()\}, \\ ....., \\ \{decision_n, a_n()\} \end{Bmatrix} \qquad (4.11)$$

In addition to these relations, there are some reserved definitions such as start and end. If they are in capital letters (START, END), they represent the start and end of a project. When they are in small letters, they represent start and end of a specific activity within the project.

D. GRAPHICAL NOTATION FOR THE MODELING LANGUAGE

It is often quoted that "*A picture is worth a thousand words*" [11]. The UML's success is a good example on the importance of graphical modeling languages. In order to ease the implementation of the proposed modeling language, it is supported with a graphical notation. This notation provides a graph-based modeling tool for practitioners and researchers. By using the notation, it possible to draw diagrams of models of project management.

A rectangle represents an activity, while an ellipse represents an entity. Figure 21 shows the graphical notation for an activity and an entity.

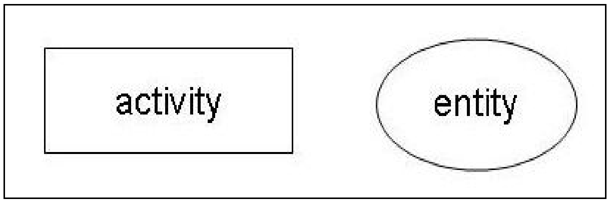


Figure 21. Graphical notation for an entity and an activity

The next and previous relations between activities are represented with an arrow. Figure 22 presents different combinations for these relations.

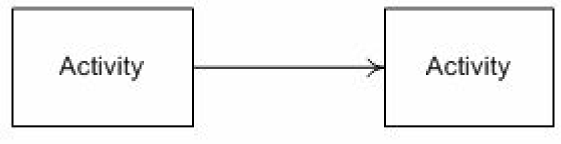


Figure 22. Next / Previous relation

Inputs to an activity are entities that are used as a reference for the execution of the activity. Inputs are connected to the activities with a line ending with a blank circle at the activity. Outputs or deliverables of an activity are entities that are produced as a result or product of an activity. Outputs are connected to the activities with a line ending at the activity with a black circle. Figure 23 represents inputs and outputs of activities.

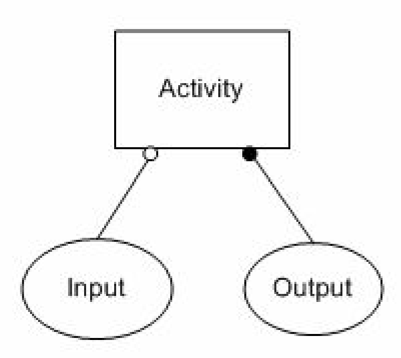


Figure 23. Inputs and Outputs

Entities like stakeholders that are related to the activities or required by the activity but do not enter to the development activity as an input to be processed, are represented by ellipses and connected to the activities with solid lines. Figure 24 represents an example for this relation.

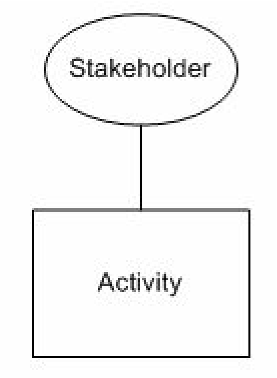


Figure 24. Stakeholder / Activity relation

In order to conduct a formal analysis, we have to provide a concept such as indivisibility. Otherwise, the hierarchy can be endless. Whenever we decide an activity or an entity should not be divided anymore, we denote indivisibility by a line under the activity name. Note that this introduces another rule to the formalism, which is a natural rule. It also represents activities and entities and requires textual explanations at this point. Figure 25 represents indivisibility.

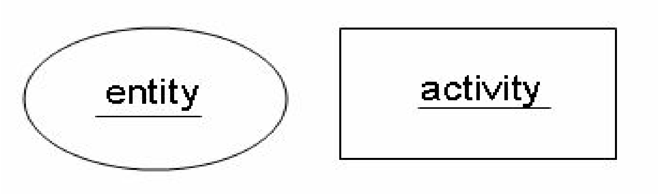


Figure 25. Indivisibility

A project start is represented by an empty small circle. A project end is represented by a full small circle. Activity and entity starts are represented by two empty circles one inside the other. Activity and entity completions are represented with a thick black circle with an empty center. Figure 26 represents start and end of project and sub-phases.

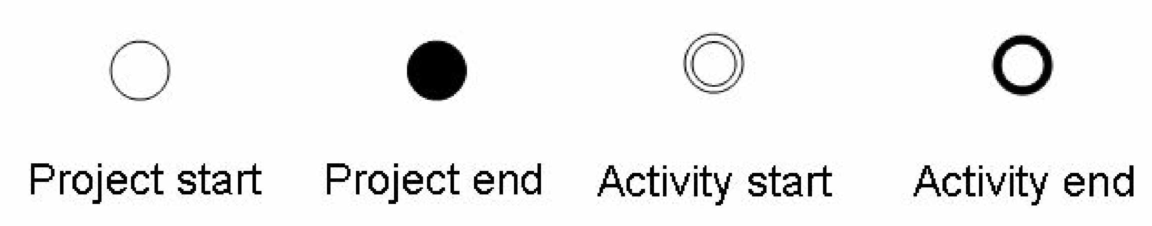


Figure 26. Start / End Relations

THIS PAGE INTENTIONALLY LEFT BLANK

V. SOUNDSTAGE ENTERTAINMENT CLUB CASE STUDY

This chapter discusses the theory and the modeling language based on a case study. The case study is taken from “System Analysis and Design” book written by Jeffery L. Whitten, Lonnie D. Bently, and Kevin C. Dittman [22]. The case study follows a waterfall-like methodology which authors call “FAST Development Methodology” to build a software system for a commercial organization named SoundStage Entertainment Club. FAST stands for Framework for the Application Systems Techniques. It consists of a sequence of steps following each other beginning with the project initiation and ending at delivery of the operational system.

This chapter does not recite the case study. This chapter models the management of the SoundStage project with the modeling language and then interprets the project management based on the model. The goal is to analyze the capability of the model to represent the project management.

In the model, we assigned symbols to activities and entities. The purpose of these assignments is only to abbreviate and simplify the mathematical model. Actual names of activities and entities can also be used instead of these abbreviations. The names of the activities and entities are substituted with these abbreviations.

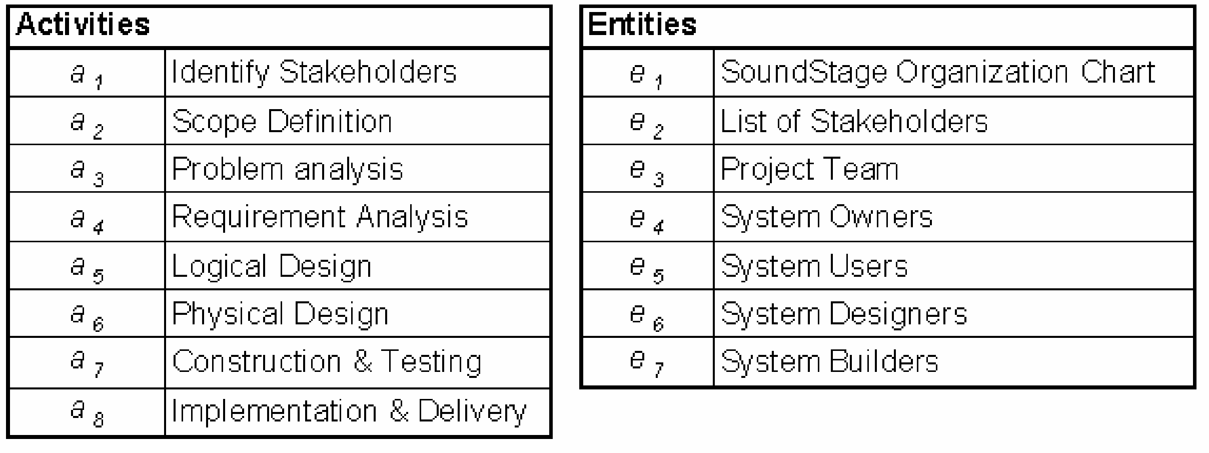
When an activity or entity is defined and added to the model, it’s scope is global. Even though it may be used in different levels of the model, the activity or entity refers to same thing. For example, activity “Build a Prototype (a_n()) refers to the same activity at all levels of the project.

A. SOUNDSTAGE ENTERTAINMENT CLUB

SoundStage Entertainment Club is a commercial organization that sells music, video products and accessories. The company is a nationwide organization with warehouses and sales offices in different states. The company plans to renew the member services information system that will affect marketing, subscriptions, sales and order entry, warehousing, inventory control and procurement, shipping and receiving, member services, suppliers, etc. The goal of the project is defined as developing new business processes and supporting information system processes and services to support the vision for SoundStage products and member services. The company plans to increase their market share with the new system and serve their members online. The project will build a network and internet based system to follow personnel issues, products, stocks, warehouses, and members via network and online.

1. Highest Level Project Model

As mentioned above, the project team follows a waterfall-like methodology called "FAST". Mathematical model of the model is as follows:



Activities	
a_1	Identify Stakeholders
a_2	Scope Definition
a_3	Problem analysis
a_4	Requirement Analysis
a_5	Logical Design
a_6	Physical Design
a_7	Construction & Testing
a_8	Implementation & Delivery

Entities	
e_1	SoundStage Organization Chart
e_2	List of Stakeholders
e_3	Project Team
e_4	System Owners
e_5	System Users
e_6	System Designers
e_7	System Builders

Table 2. SoundStage Project Highest Level Model Activities and Entities

Stakeholders of the project comprise five separate groups. The project team is responsible for controlling, managing and executing the project. System Owners are the upper-managers of the requester organization responsible for monitoring and approving the project. System designers are system analysts responsible for designing the software structure of the project. System users are end-users who will actually use the final product.

General formulation includes main activities (***a***) and entities (***e***) in the project. However, this does not mean that there are no other activities or entities in the project. Sub-activities and entities will be derived form the general formula using divide, aggregate, create, etc. relations.

$$\begin{aligned} P_{ss} = PM\{&START, a_1(), a_2(), a_3(), a_4(), a_5(), a_6(), a_7(), a_8(), \\ &e_1, e_2, e_3, e_4, e_5, e_6, e_7, END\} \end{aligned}$$

$$\boldsymbol{\begin{aligned} &P_{ss} = PM\{ START, identify\_stakeholders, scope\_definition, problem\_analysis, \\ &requirement\_analysis, logical\_design, physical\_design, construction\&testing, \\ &implementation\&delivery, soundstage\_organization\_chart, list\_of\_stakeholders, \\ &project\_team, system\_owners, system\_users, system\_designers, system\_builders, END\} \end{aligned}}$$

We use (a_1) and (e_1) as a choice. Instead of (a_1) and (e_1), (a_{is}) and (e_{soc}) can also be used as notation.

Here, the formulation states that SoundStage project's project management includes activities $a_1()$, $a_2()$, $a_3()$, $a_3()$, $a_4()$, $a_5()$, $a_6()$, $a_7()$, $a_8()$ and entities e_1, e_2, e_3, e_4, e_5, e_6, e_7.

Relations between activities are as follows:

$$START \rightarrow a_1(\ )$$

$$a_1(\ ) \rightarrow a_2(\ )$$

$$a_2(\ ) \rightarrow a_3(\ )$$

$$a_3(\ ) \rightarrow a_4(\ )$$

$$a_4(\ ) \rightarrow a_5(\ )$$

$$a_5(\ ) \rightarrow a_6(\ )$$

$$a_6(\ ) \rightarrow a_7(\ )$$

$$a_7(\,) \to a_8(\,)$$

$$a_8(\,) \to END$$

The first activity of the project is “Identify Stakeholders (a_1)” activity. Project comprises eight consecutive activities from start to end.

Relations between activities, inputs and outputs are as follows.

$$e_2 = a_1(e_1)$$

Relation above says “SoundStage Organization Chart (e_1)” entity transforms into “List of Stakeholders (e_2)” entity as a result of “Identify stakeholders (a_1)” activity. It states “Identify Stakeholders” activity creates “List of Stakeholders” entity.

Stakeholders, their responsibilities and relations to any activity or entity can be formulated using “Require” relation.

$$REQUIRE(a_1(\,)) = \{e_3\}$$

$$REQUIRE(a_2(\,)) = \{e_3, e_4\}$$

$$REQUIRE(a_3(\,)) = \{e_3, e_5\}$$

$$REQUIRE(a_4(\,)) = \{e_3, e_5\}$$

$$REQUIRE(a_5(\,)) = \{e_3, e_5\}$$

$$REQUIRE(a_6(\,)) = \{e_3, e_5, e_6\}$$

$$REQUIRE(a_7(\,)) = \{e_3, e_5, e_7\}$$

$$REQUIRE(a_8(\,)) = \{e_3, e_5, e_6\}$$

The project management is modeled below in Figure 27:

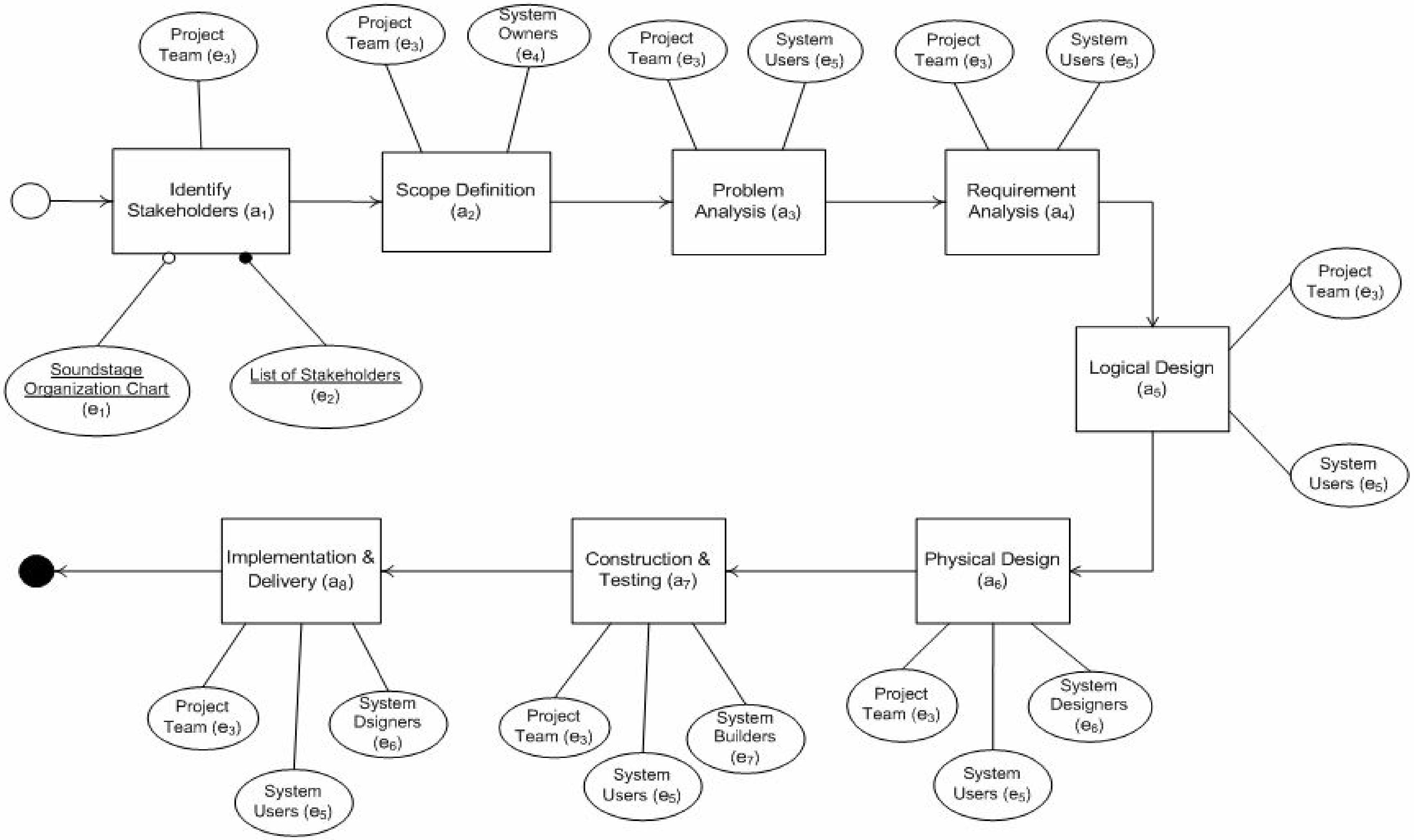


Figure 27. Soundstage Entertainment Club Highest Level Project Plan

Let us explain the meaning of graphical representations in the model. We will focus on the relations of the “Identify Stakeholder” activity as an example.

The blank circle in the figure depicts the start node of the project and black circle represents the end node of the project. As described in Chapter IV rectangles represent activities and ellipses represent entities such as inputs, outputs and stakeholders.

The first activity of the project is the identification of the stakeholders. The text inside the activity rectangle is not underlined. This means that “Identify Stakeholders” activity is composed of subtasks. Those subtasks will further be modeled until it reaches a point where we do not need to detail anymore. Details about activities and entities may be explained in plain English if needed. The activity that follows “Identify Stakeholders” activity is “Scope Definition”. Activities in the model are connected with an arrow-ended line to depict next and previous relationships. The project comprises activities starting with identify stakeholders, continues with scope definition, problem analysis, requirements analysis, logical design, physical design, construction & testing activities in sequence and ends with implementation and delivery of the operational system. The activities are not underlined. Therefore, they have subtasks and activities that require further modeling. When an activity or entity is underlined it means there is no further need to graphically model the activity or entity.

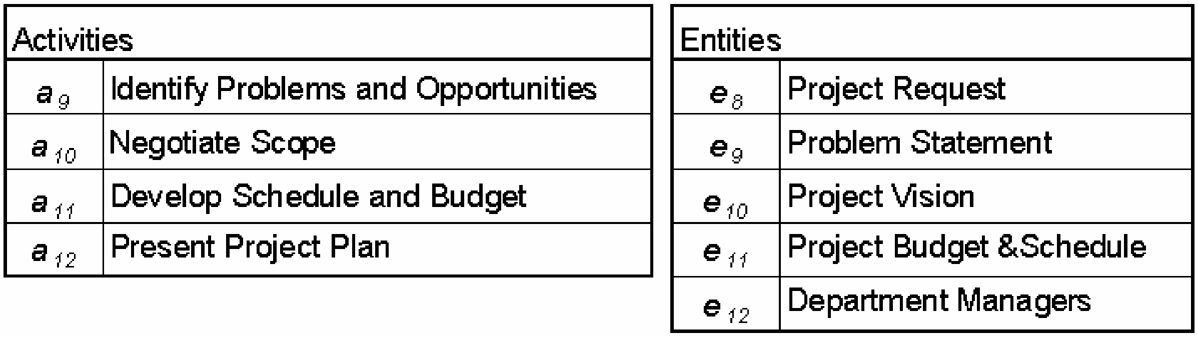
The ellipse tied to the “Identify Stakeholders” activity with a solid line depicts the stakeholders who perform or participate in the activity. The text in the box is not underlined. This shows that there are sub-groups of project team. The model further depicts these sub-groups and their participation to sub-activities and tasks. Solid lines connect the stakeholders to the activities they participate. For example, the stakeholders who perform or be involved in “Physical Design” activity comprises project team, system users and system designers.

The ellipse tied to “Identify Stakeholder” activity with a line ending with a empty small circle depicts that SoundStage company organization chart is the input to “Identify Stakeholder” activity. All the tasks and subtasks in this activity are executed based on the specifications or schemas defined in the organization chart. The performers of the activity will use the chart as a basis in performing the “Identify Stakeholder” activity.

The output of stakeholder definition activity is depicted with an ellipse that is tied to the activity with a line starting with a black small circle at the activity. This shows that “list of stakeholders” is the output of “Identify Stakeholder”. The text inside the ellipse is underlined. The entity may be detailed with a plain text.

2. Scope Definition Phase

We are diving further into the details of the project. The highest-level model is described above. Let us continue with the “Scope Definition” phase. Mathematical model of the “Scope Definition’ phase is below:



Activities	
a_9	Identify Problems and Opportunities
a_{10}	Negotiate Scope
a_{11}	Develop Schedule and Budget
a_{12}	Present Project Plan

Entities	
e_8	Project Request
e_9	Problem Statement
e_{10}	Project Vision
e_{11}	Project Budget &Schedule
e_{12}	Department Managers

Table 3. SoundStage Project Scope Definition Phase Activity / Entity List

General formula for scope definition phase is:

$$DIVIDE(a_2(\,)) = \{\, start_{a_2}, a_9(\,), a_{10}(\,), a_{11}(\,), a_{12}(\,), e_3, e_4, e_8, e_9, e_{10}, e_{11}, e_{12}, end_{a_2} \}$$

DIVIDE(scope_definition) = { start $_{a_2}$ ***,identify_problems_and_opportunities, negotiate_scope,develop_schedule_and_budget, present_project_plan, project_request, problem_statement,project_vision,project_budget_&_schedule, department_managers, end*** $_{a_2}$ ***}***

Scope Definition activity (***a**2*) comprises ***a**9,* ***a**10,* ***a**11,* ***a**12* activities and ***e**3,* ***e**4,* ***e**8,* ***e**9,* ***e**10,* ***e**11,* ***e**12* entities.

Previous/Next relations between activities are:

$$start_{a_2} \rightarrow a_9(\,)$$

$$a_9(\,) \rightarrow a_{10}(\,)$$

$$a_{10}(\,) \rightarrow a_{11}(\,)$$

$$a_{11}(\,) \rightarrow a_{12}(\,)$$

$$a_{12}(\,) \rightarrow end_{a_2}$$

The "Scope Definition" phase starts with "Identify Problems and Opportunities" activity (***a**9*). The activities are subsequent. There are previous and next relationships between them. ***a**10* follows ***a**9* , ***a**11* follows ***a**10* ,***a**12* follows ***a**11,* and the phase ends with "Present Project Plan (***a**12*)" activity.

Relations between activities, inputs and outputs are:

$$e_9 = a_9(e_8)$$

$$e_{10} = a_{10}(e_9)$$

$$e_{11} = a_{11}(e_{10})$$

$$REQUIRE(a_{12}(\,)) = \{e_{11}\}$$

The relations above identify transform relations between activities and entities. "Project Request (***e**8*)" entity is used as an input to "Identify Problems and Opportunities (***a**9*)" activity. Output of the activity is "Problem Statement (***e**8*)". ***e**9* is used as an input to ***a**10* and the output is ***e**10*. ***e**10* is used as an input to ***a**11* and the output is ***e**11*. There is also a require relation between ***e**11* and ***a**12*.

“Project Budget and Schedule” is required for execution of “Present Project Plan” activity.

Relations of stakeholders to activities are as follows:

$$REQUIRE(a_9(\,)) = \{e_3, e_4, e_{12}\}$$
$$REQUIRE(a_{10}(\,)) = \{e_3, e_4, e_{12}\}$$
$$REQUIRE(a_{11}(\,)) = \{e_3\}$$
$$REQUIRE(a_{12}(\,)) = \{e_3\}$$

Activities $\boldsymbol{a_9}$ and $\boldsymbol{a_{10}}$ will be performed by $\boldsymbol{e_3}$, $\boldsymbol{e_4}$ and $\boldsymbol{e_{12}}$, or they will participate in these activities. Project team will perform activities $\boldsymbol{a_{11}}$ and $\boldsymbol{a_{12}}$.

Figure 28 displays the model of the scope definition phase of the project.

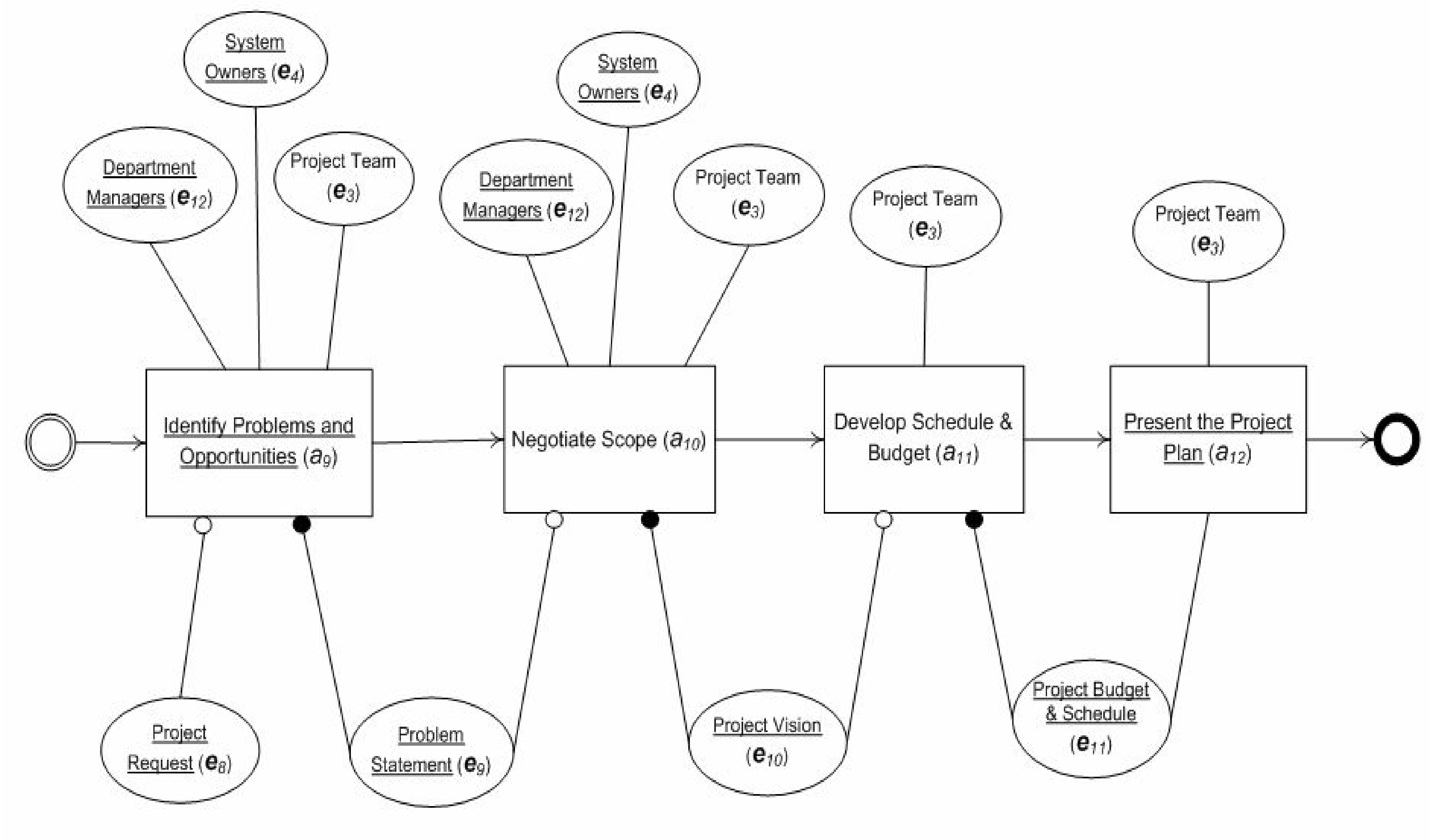

System Owners (e_4)
Department Managers (e_{12})
Project Team (e_3)
System Owners (e_4)
Department Managers (e_{12})
Project Team (e_3)
Project Team (e_3)
Project Team (e_3)
Identify Problems and Opportunities (a_9)
Negotiate Scope (a_{10})
Develop Schedule & Budget (a_{11})
Present the Project Plan (a_{12})
Project Request (e_8)
Problem Statement (e_9)
Project Vision (e_{10})
Project Budget & Schedule (e_{11})

Figure 28. Soundstage Entertainment Club Scope Definition Phase

The Scope Definition phase is a sub-activity in the highest-level model of Sound Stage Entertainment Club Project and comprises other sub-activities and entities. Since “Scope Definition” activity is a sub-activity and detailed with a “Divide” relation, the start node is composed of two circles with a blank center, and end node is a thick black circle with a blank center.

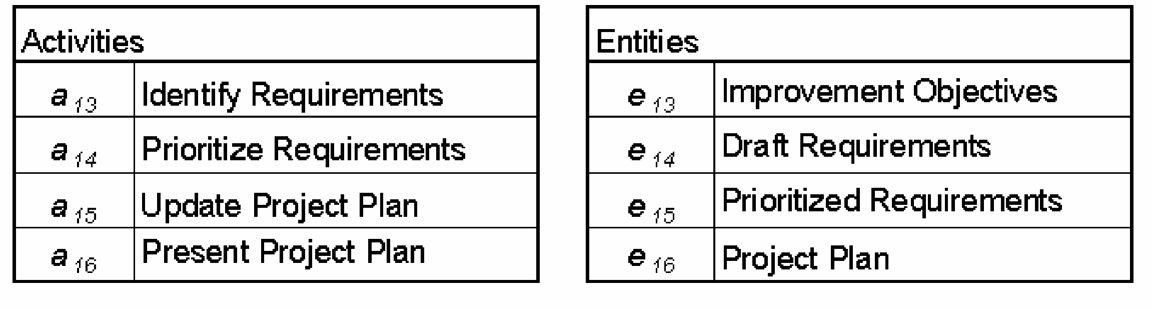
The Scope Definition phase consists of four activities following one other. Phase starts with “Identify Problems and Opportunities” activity. Following activities are “Negotiate Scope”, “Develop Schedule and Budget”, and “Present Project Plan” activities in sequence. “Negotiate Scope” and “Develop Schedule and Budget” activities have sub-activities and tasks. They require further modeling. “Identify Problems and Opportunities” and “Present the Project Plan” activities do not have sub-tasks or activities. Therefore, if needed additional texts may include definition of what and how needs to be done.

Input to “Identify Problems and Opportunities” activity is “Project Request” entity. Key deliverable of the activity is “Problem Statement” entity. Both “Problem Statement” and “Project Request” entities are underlined. Underlining means that they are not further modeled. “Project Request” is a memorandum of authority from organization requesting systems’ development. Problem Statement covers problems, opportunities and directives identified in the activity.

As the Figure 28 displays, project team participates in each activity. Project team, department managers and owners participate in both “Identify Problems and Opportunities” and “Negotiate Scope” activities. Execution of the latter two activities is the responsibility of the project team.

3. Requirements Analysis Phase

The fourth phase of the SoundStage entertainment club project is “Requirements Analysis” phase, which is critical to any software project. Mathematical model of the phase is below:



Activities	
a_{13}	Identify Requirements
a_{14}	Prioritize Requirements
a_{15}	Update Project Plan
a_{16}	Present Project Plan

Entities	
e_{13}	Improvement Objectives
e_{14}	Draft Requirements
e_{15}	Prioritized Requirements
e_{16}	Project Plan

Table 4. SoundStage Project Requirement Analysis Phase Activity / Entity List

General formula for requirement analysis phase is:

$$DIVIDE(a_4()) = \{ start_{a_4}, a_{13}(), a_{14}(), a_{15}(), a_{16}(), e_3, e_4, e_5, e_{13}, e_{14}, e_{15}, e_{16}, end_{a_4} \}$$

Requirement Analysis activity (***a***$_4$) comprises ***a***$_{13,}$ ***a***$_{14,}$ ***a***$_{15,}$ ***a***$_{16}$ activities and ***e***$_{3,}$ ***e***$_{4,}$ ***e***$_{5,}$ ***e***$_{13,}$ ***e***$_{14,}$ ***e***$_{15,}$ ***e***$_{16}$ entities.

Relations between activities are:

$$start_{a_4} \to a_{13}(\,)$$

$$a_{13}(\,) \to a_{14}(\,)$$

$$a_{14}(\,) \to a_{15}(\,)$$

$$a_{15}(\,) \to a_{16}(\,)$$

$$a_{16}(\,) \to end_{a_4}$$

Requirement Analysis phase starts with “Identify Requirements (a13)” activity. The activities are subsequent and the phase ends with “Present Project Plan (***a***$_{16}$)” activity.

Relations between activities, inputs and outputs are:

$$e_{14} = a_{13}(e_{13})$$

$$e_{15} = a_{14}(e_{14})$$

$$e_{16} = a_{15}(e_{15})$$

$$REQUIRE(a_{16}()) = \{e_3, e_{16}\}$$

"Improvement Objectives (***e**13*)" entity is an input to "Identify Requirements (a13)" activity. Output of the activity is "Draft Requirements (***e**14*)". ***e**14* is used as an input to ***a**14* and the output is ***e**15*. ***e**15* is used as an input to ***a**15* and the output is ***e**16*. e_3 and ***e**16* are required for ***a**16* activity.

Relations of stakeholders to any activities are:

$$REQUIRE(a_{13}()) = \{e_3, e_5\}$$

$$REQUIRE(a_{14}()) = \{e_3, e_4, e_5\}$$

$$REQUIRE(a_{15}()) = \{e_3\}$$

Project team (***e**3*) will be involved in all of the activities. Owners will be included in ***a**15*.. System users will participate in activities ***a**13* and ***a**14*.

Figure 29 shows the graphical model of the phase.

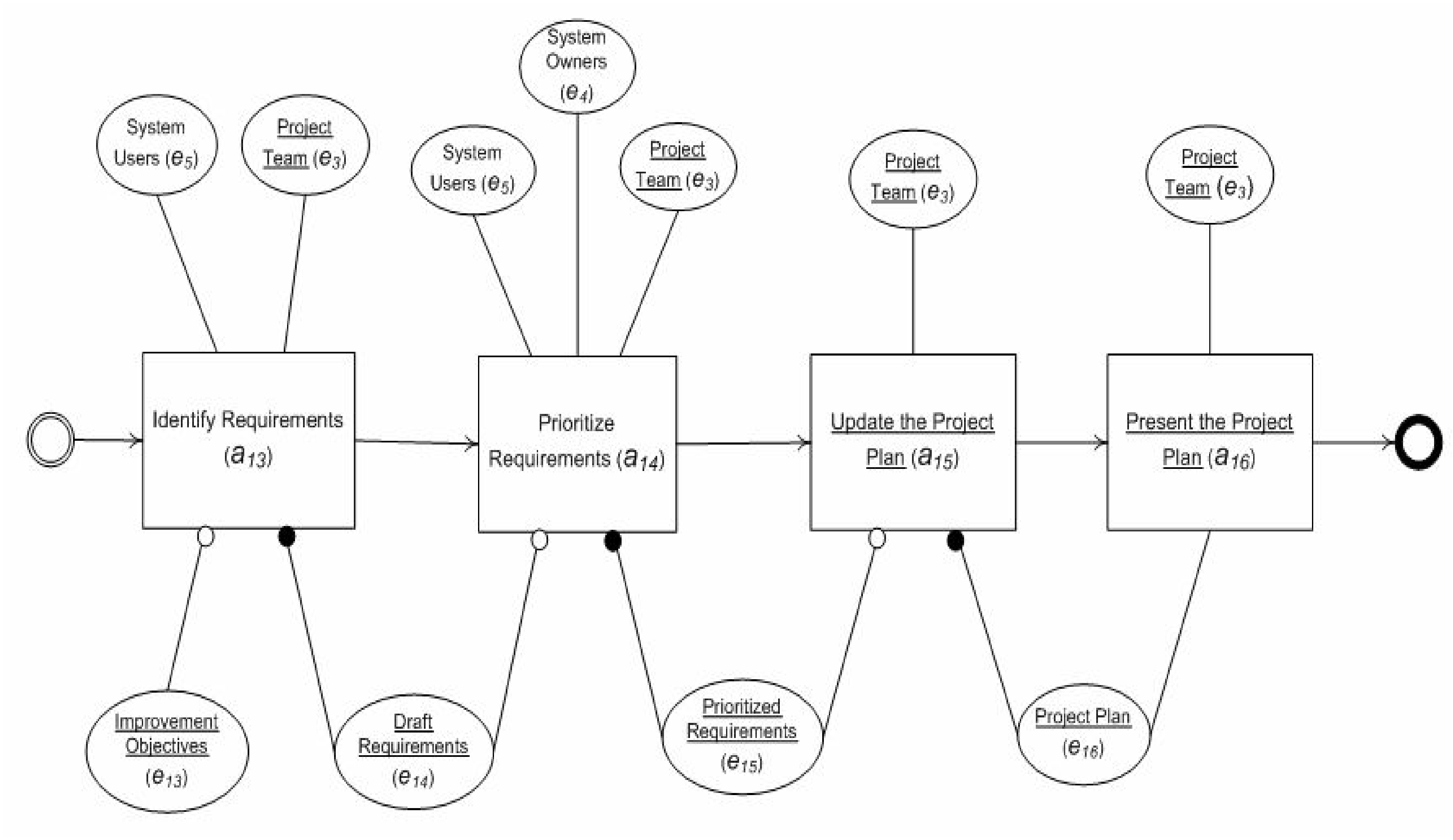

System Owners (e_4)
System Users (e_5)
Project Team (e_3)
System Users (e_5)
Project Team (e_3)
Project Team (e_3)
Project Team (e_3)
Identify Requirements (a_{13})
Prioritize Requirements (a_{14})
Update the Project Plan (a_{15})
Present the Project Plan (a_{16})
Prioritized Requirements (e_{15})
Project Plan (e_{16})
Draft Requirements (e_{14})
Improvement Objectives (e_{13})

Figure 29. SoundStage Project Requirements Analysis Phase

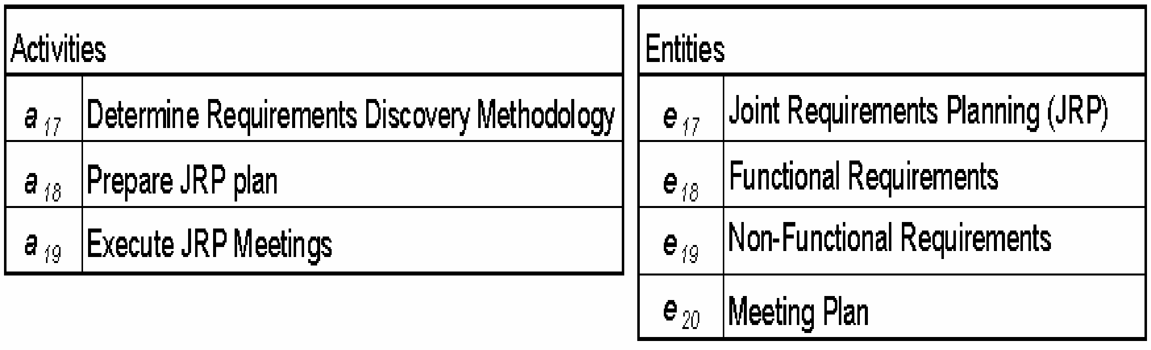
Requirements Analysis phase of the project comprises four consecutive activities. This phase starts with “Identify Requirements” activity; it continues with “Prioritize Requirements”, “Update the Project Plan” activities; and it ends with “Present the Project Plan” activity. The first two activities have subtasks. The latter two activities are not modeled further.

The “Improvement Objectives” entity is the input to the “Identify Requirements” activity. The performers of the activity take “Improvement Objectives” as a basis when they execute the process. Improvement Objectives are determined in the “Problem Analysis” phase of the project that is the preceding phase to the “Requirements Analysis” phase.

The key deliverable or output of the “Identify Requirements” activity is the “Draft Requirements” entity and it is an input to the following “Prioritize Requirements” activity.

a. *“Identify Requirements” Activity*

As mentioned previously, “Identify Requirements” activity has sub-activities that have graphical representations in the model. Mathematical expression of the model is:



Activities	
a_{17}	Determine Requirements Discovery Methodology
a_{18}	Prepare JRP plan
a_{19}	Execute JRP Meetings

Entities	
e_{17}	Joint Requirements Planning (JRP)
e_{18}	Functional Requirements
e_{19}	Non-Functional Requirements
e_{20}	Meeting Plan

Table 5. SoundStage Project Identify Requirements Activity / Entity List

General relation for requirement analysis phase is:

$$DIVIDE(a_{13}()) = \{ start_{a_{13}}, a_{17}(), a_{18}(), a_{19}(), e_3, e_5, e_{13}, e_{17}, e_{18}, e_{19}, e_{20}, end_{a_{13}} \}$$

Identify Requirements activity (***a***$_{13}$) comprises ***a***$_{17,}$ ***a***$_{18,}$ ***a***$_{19}$ activities and ***e***$_{3,}$ ***e***$_{5,}$ ***e***$_{13,}$ ***e***$_{17,}$ ***e***$_{18,}$ ***e***$_{19,}$ ***e***$_{20}$ entities.

Relations between activities are:

$$start_{a_{13}} \to a_{17}(\ )$$

$$a_{17}(\ ) \to a_{18}(\ )$$

$$a_{18}(\ ) \to a_{19}(\ )$$

$$a_{19}(\ ) \to end_{a_{13}}$$

"Identify Requirements" activity comprises three subsequent sub-activities. Tasks start with "Determine Requirements Discovery Methodology (***a***$_{17}$)" activity, continue with "Prepare JRP Plan" activity and ends with "Execute JRP meetings (***a***$_{19}$)" activity.

Relations between activities, inputs and outputs are:

$$e_{17} = a_{17}(e_{13})$$

$$e_{20} = a_{18}(e_{17})$$

$$e_{18} = a_{19}(\ \ )$$

$$e_{19} = a_{19}(\ \ )$$

The "Improvement Objectives (***e***$_{13}$)" entity is an input to "Determine Requirements Discovery Methodology (***a***$_{17}$)" activity. Output of the activity is "JRP (***e***$_{17}$)". ***e***$_{17}$ is used as an input to ***a***$_{18}$ and the deliverable is ***e***$_{20}$. ***a***$_{19}$ creates ***e***$_{18}$ and ***e***$_{19}$ as outputs.

Relations of stakeholders to activities are:

$$REQUIRE(a_{17}(\,)) = \{e_3\}$$

$$REQUIRE(a_{18}(\,)) = \{e_3\}$$

$$REQUIRE(a_{19}(\,)) = \{e_3, e_5\}$$

The project team (***e**$_3$*) performs all of the activities. System Users participate in JRP meetings (***a**$_{19}$*) to determine desired requirements for the final product of the project.

Figure 30 shows the model of the activity.

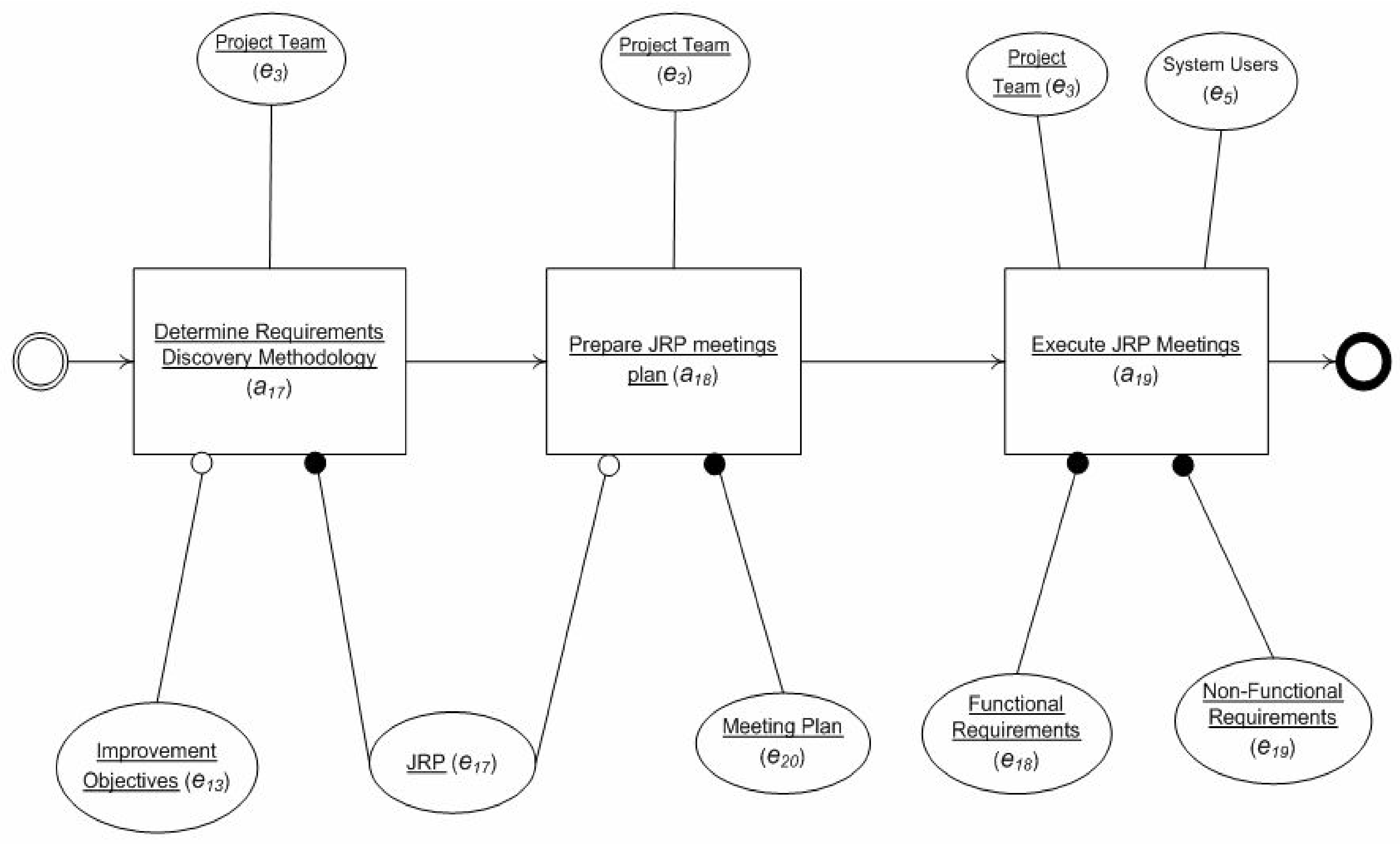
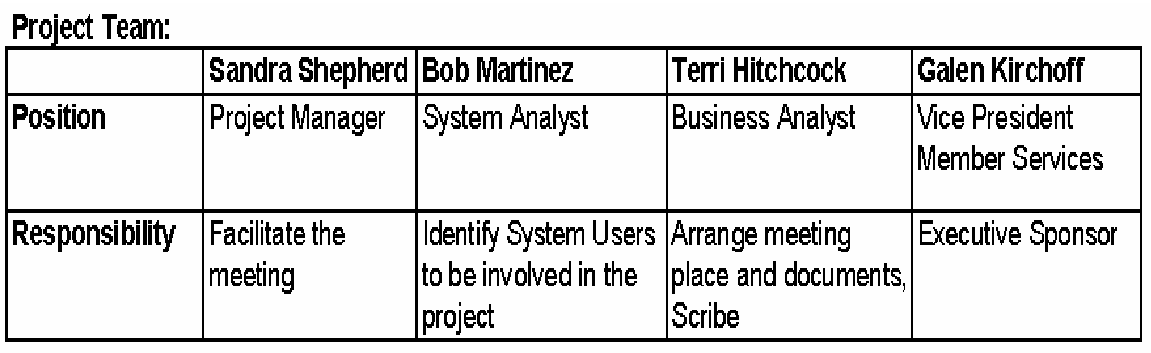

Project Team (e_3)
Project Team (e_3)
Project Team (e_3)
System Users (e_5)
Determine Requirements Discovery Methodology (a_{17})
Prepare JRP meetings plan (a_{18})
Execute JRP Meetings (a_{19})
Improvement Objectives (e_{13})
JRP (e_{17})
Meeting Plan (e_{20})
Functional Requirements (e_{18})
Non-Functional Requirements (e_{19})

Figure 30. SoundStage Project Identify Requirements Activity

“Identify Requirements” activity comprises three activities. These three activities are consequent sub-activities and start with determining requirement discovery methodology, continue with preparing Joint Requirement Planning (JRP) meetings plan and end with JRP meetings.

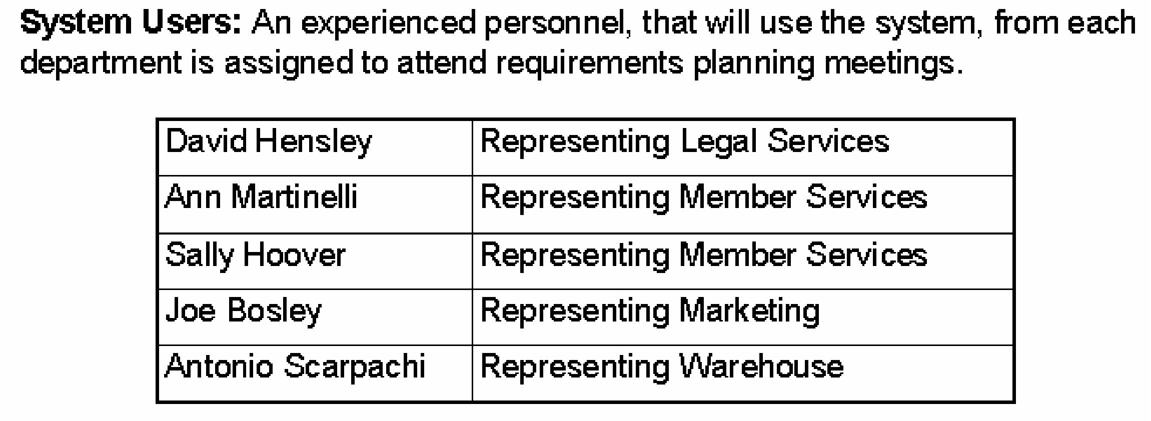
The project team executes “Determine Requirements Discovery Methodology” and “Prepare JRP Meetings Plan” activities. System users who will use the final product are involved in JRP meetings along with the project team to determine requirements for the system. “Project Team” and “System User” entities are underlined and they will not be further modeled. Table 6 and Table 7 display the textual explanations of project team, system users and their roles.



Project Team:

	Sandra Shepherd	**Bob Martinez**	**Terri Hitchcock**	**Galen Kirchoff**
Position	Project Manager	System Analyst	Business Analyst	Vice President Member Services
Responsibility	Facilitate the meeting	Identify System Users to be involved in the project	Arrange meeting place and documents, Scribe	Executive Sponsor

Table 6. SoundStage Project Team

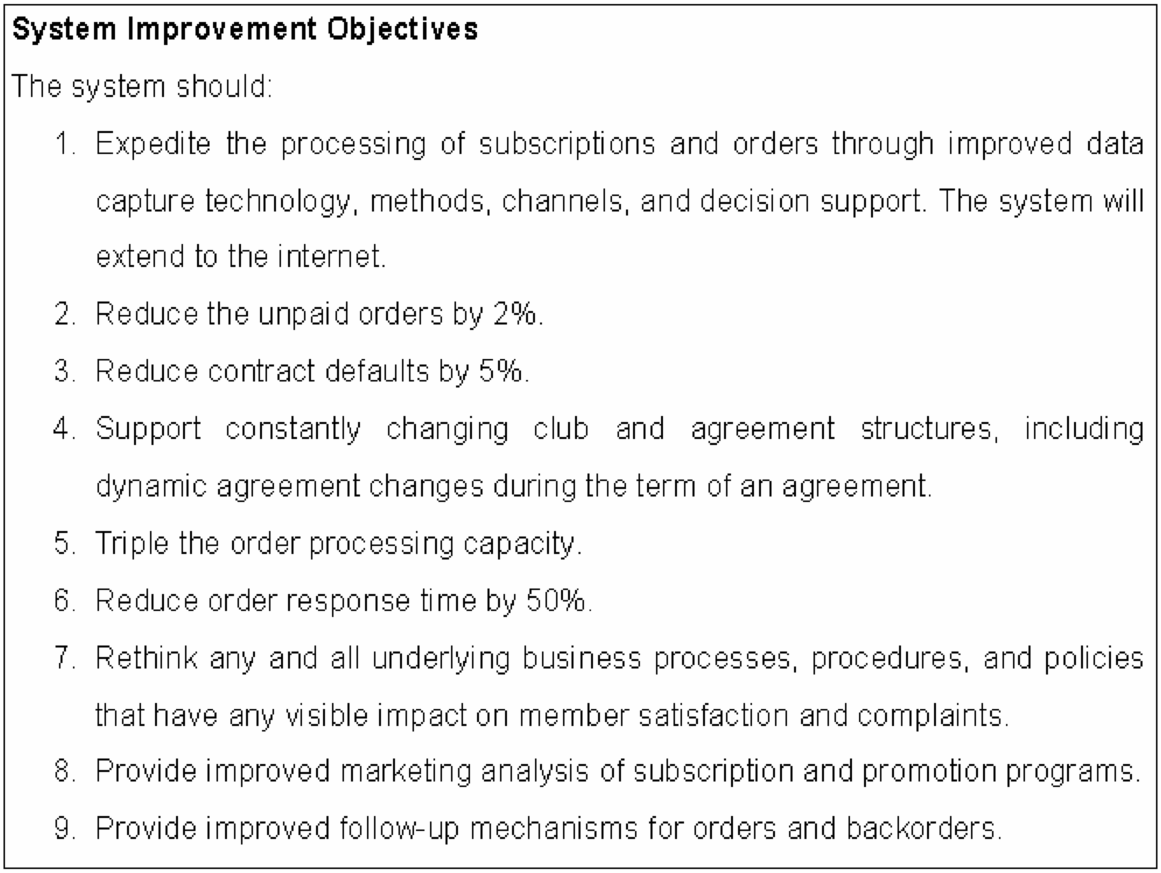


System Users: An experienced personnel, that will use the system, from each department is assigned to attend requirements planning meetings.

David Hensley	Representing Legal Services
Ann Martinelli	Representing Member Services
Sally Hoover	Representing Member Services
Joe Bosley	Representing Marketing
Antonio Scarpachi	Representing Warehouse

Table 7. SoundStage Project System Users

"System Improvement Objectives" triggers the "Identify Requirements" activity. Requirements discovery methodology is determined based on system improvement objectives (Figure 31). "System Improvement Objectives" is output of previous activities in the Problem Analysis phase of the project. Final outputs of the activities are draft functional and non-functional requirements. Functional requirements are critical requirements that should be addressed for the project success. Non-functional requirements are requirements that may be addressed based on owners' expectations or scope of the project.



System Improvement Objectives

The system should:

1. Expedite the processing of subscriptions and orders through improved data capture technology, methods, channels, and decision support. The system will extend to the internet.
2. Reduce the unpaid orders by 2%.
3. Reduce contract defaults by 5%.
4. Support constantly changing club and agreement structures, including dynamic agreement changes during the term of an agreement.
5. Triple the order processing capacity.
6. Reduce order response time by 50%.
7. Rethink any and all underlying business processes, procedures, and policies that have any visible impact on member satisfaction and complaints.
8. Provide improved marketing analysis of subscription and promotion programs.
9. Provide improved follow-up mechanisms for orders and backorders.

Figure 31. SoundStage Project System Improvement Objectives (After: [22])

We focused on the highest-level plan and some activities in the model. The entire project can be modeled in detail as above. Appendix displays the entire model of the project.

B. EVALUATION OF SOUNDSTAGE MODELS WITH DESIRED CRITERIA

1. Simplicity

The case study is modeled with very few shapes and connections that make it easy to figure out what the project is about, what it covers. The model is simple but comprehensive enough to cover a relatively high amount of information about the project.

2. Inclusion of Different Aspects of the Software Projects

The project manager has to deal with scope, budget, cost, quality, stakeholders and risk. These aspects comprise the project managers' responsibilities. The model depicting the SoundStage case study comprises scope, quality and people aspects of the project and the final product. The model covers the required quality by determining the features or specifications of the outputs of all activities and tasks. This model covers the scope of the project by determining phases of the project, inputs and outputs, and people involved. However, this model lacks cost and schedule aspects of the project.

3. Work Breakdown Structure of the Project

The model comprises all of the activities, tasks, subtasks of the project. The model also displays who performs or participates in any activity. Furthermore, model includes inputs and outputs of any activity. In essence, project team can determine the work breakdown structure of the project from the model.

4. Inputs and Outputs of the Project

Inputs and outputs determine the quality and scope of the project. They also allow project team to set criteria for any activity and determine what to expect with the completion of the activity or task. The model introduced above comprises the inputs and outputs of all of the activities in the project.

5. The Modeling Tool Permits Formal Analysis

The model of the SoundStage project allows for formal analysis. The project team can analyze, interpret and derive results from the model. The project team can see the absent or missing aspects of the project from the model. The model can be updated based on the analysis.

6. Support to Existing Project Management Methodologies

The model has abstract features that depict different aspects of software projects including stakeholders, activities, inputs, outputs and relationships between those. FAST methodology is a modified version of Waterfall lifecycle development methodology. Since accepted features of those methodologies comprise what a software development project should include and the proposed model comprises most of those features, then the model supports proven methodologies. Popular methodologies are introduced in Chapter II.

7. Stakeholders, Their Responsibilities and Relations to Activities

Stakeholders are crucial to project success. Stakeholders comprise anybody involving in the project. The model of the SoundStage project comprises all of the stakeholders participating in the project and displays responsibilities and roles of each stakeholder.

VI. F-16 CASE STUDY

This chapter is based on the F-16 Fighting Falcon case study developed by Technology Management School, Center for Professional Development, Air University [8].

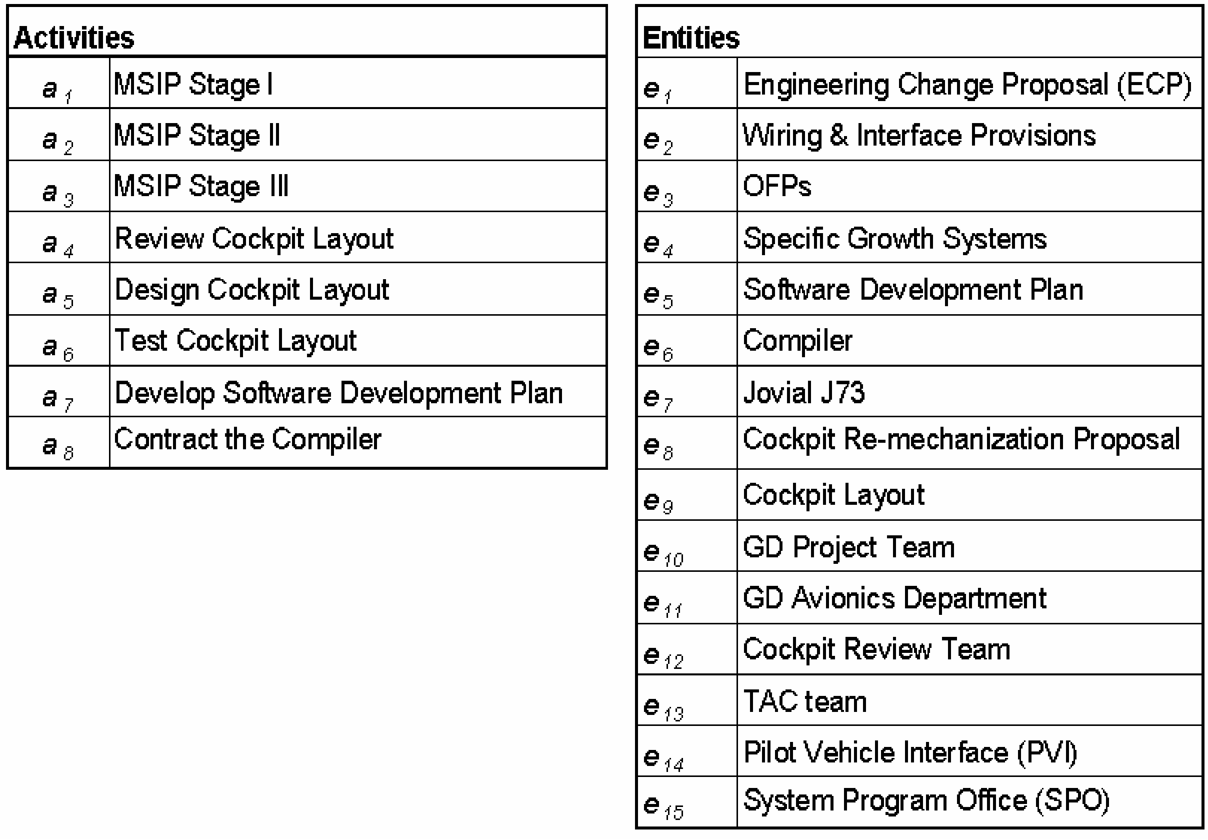
General Dynamics (GD) was awarded the contract for building a third generation aircraft for U.S. Air Force (F-16). This case study discusses software development issues of the project from managerial perspective.

Software development lifecycle of the project comprises three aspects. These three aspects are development of Operational Flight Programs (OFP) for new systems, integration of newly developed systems by other contractors, and re-mechanization of F-16 cockpit to suit newly developed systems. General Dynamics established a plan called Multi-stage Improvement Plan (MSIP) for the project. These plans are modeled.

A. F-16 SOFTWARE DEVELOPMENT PLAN

Multi-stage Improvement Plan has three subsequent phases comprising parallel OFP development, sub-system integration and cockpit re-mechanization for F-16 aircraft. MSIP Stage I provides essential structure, wiring and interface provisions to support future avionic changes and new growth systems. MSIP Stage II provides new aircraft avionics and sub-system improvements to support necessary future growth. MSIP Stage III provides for installation and retrofit of future growth systems as well as integration of sub-systems developed by different contractors.

Activities and entities which project comprise are listed below:



Activities	
a_1	MSIP Stage I
a_2	MSIP Stage II
a_3	MSIP Stage III
a_4	Review Cockpit Layout
a_5	Design Cockpit Layout
a_6	Test Cockpit Layout
a_7	Develop Software Development Plan
a_8	Contract the Compiler

Entities	
e_1	Engineering Change Proposal (ECP)
e_2	Wiring & Interface Provisions
e_3	OFPs
e_4	Specific Growth Systems
e_5	Software Development Plan
e_6	Compiler
e_7	Jovial J73
e_8	Cockpit Re-mechanization Proposal
e_9	Cockpit Layout
e_{10}	GD Project Team
e_{11}	GD Avionics Department
e_{12}	Cockpit Review Team
e_{13}	TAC team
e_{14}	Pilot Vehicle Interface (PVI)
e_{15}	System Program Office (SPO)

Table 8. F-16 Project Highest Level Activity / Entity List

Mathematical model of the highest-level project is as follows:

$$P_{f16}=PM\{START,a_1(),a_2(),a_3(),a_4(),a_5(),a_6(),a_7(),a_8(),e_1,e_2,e_3,\\ e_4,e_5,e_6,e_7,e_8,e_9,e_{10},e_{11},e_{12},e_{13},e_{14},e_{15},END\}$$

F-16 project comprises eight activities from a1 to a8 and fifteen entities from e1 to e15. Entities listed above include inputs, outputs and stakeholders of the project.

Relations between activities are as follows:

$$START \rightarrow a_1(\,)$$

$$START \rightarrow a_7(\,)$$

$$START \rightarrow a_8(\,)$$

$$a_1(\,) \rightarrow a_2(\,)$$

$$a_2(\,) \rightarrow a_3(\,)$$

$$a_2(\,) \rightarrow a_4(\,)$$

$$a_4(\,) \rightarrow a_5(\,)$$

$$a_5(\,) \rightarrow a_6(\,)$$

$$a_7(\,) \rightarrow a_2(\,)$$

$$a_3(\,) \rightarrow END$$

$$a_6(\,) \rightarrow END$$

The relations above determines the sequence of activities and previous, next relationships between them. The MSIP project starts with three parallel activities $a_{1,}$ $a_{7,}$ and a_8 separately. The phase starting with “MSIP Stage I (a_1)”, continues with a_2. Another parallel phase starts with “Develop Software Development Plan (a_7)” and combines with “MSIP Stage II (a_2)”. The last parallel activity is “Contract the Compiler (a_8)” for source codes in order to make them executable programs. Compiler is an input to “MSIP Stage II (a_2)”.

“MSIP Stage II (a_2)” has two parallel follow-on activities. The new systems developed in a_2 results in a need for re-mechanization of F-16 cockpit to integrate the new systems. Cockpit re-mechanization activities start with “Review Cockpit Layout (a_4)”, continues with “Design Cockpit Layout (a_5)” and ends with “Testing Cockpit Layout (a_6)”. Another follow-on activity of a_2 is “MSIP Stage III (a_3)”. The software project ends with the completion of a_3 and a_6.

Relations between activities, inputs and outputs are as follows:

$$e_2 = a_1(e_1)$$

$$e_5 = a_7(\ )$$

$$e_6 = a_8(e_7)$$

$$e_3 = a_1(e_2, e_5, e_6)$$

$$e_4 = a_3(\ )$$

$$e_8 = a_4(\ )$$

$$e_9 = a_5(e_8)$$

$$e_{14} = a_6(e_3, e_4, e_9)$$

Performers of “MSIP Stage I (a_1)” use “ECP (e_1)” as an input to execute the tasks. The deliverable or output of the activity is “Wiring & Interface Provisions (e_2)” for new systems of F-16.

The General Dynamics project team prepares a “Software Development Plan (e_5)” for software issues of the ongoing project including budget, schedule and follow-on activities.

The Air Force requires the software to use “Jovial J73 (e_7)” language for the systems. General Dynamics signs a contract (a_8) with an outside contractor to develop the required compiler (e_6) that will satisfy the requirements for Jovial J73.

MSIP Stage II (a_2) is where software for new avionic systems is developed. The requirements for new software are determined in “wiring & interface provisions (e_2)”, “software development plan (e_5)” and the “compiler

(e_6)" is used to transform written codes into executable programs. The deliverable or output of MSIP Stage II is approved "operational flight programs (e_3)" for the new systems.

"MSIP Stage III (a_3)" is a follow-on activity to MSIP Stage II. In Stage III outsourced "Specific Growth Systems (e_4)" like AMRAAM (Advanced Middle Range Air to Air Missile), ECM (Electronic Counter Measures), etc, are integrated to F-16 avionics to improve F-16's war-fighting capability by General Dynamics Avionics Department.

A parallel activity of the project is the requirement for cockpit re-mechanization to fit the newly developed systems. In order to design a new cockpit layout, the cockpit review team examines the existing layout (a_4) and prepares a cockpit re-mechanization proposal (e_8) to be approved by the U.S. Air Force. The re-mechanization proposal is the base or input for designing the new cockpit layout (a_5) activity. The deliverable of the a_5 activity is the new cockpit layout design (e_9).

Systems developed with new OFPs (e_3) on MSIP Stage II and integrated specific growth systems (e_4) on MSIP Stage III are continuously tested (a_6) with the new cockpit layout (e_9). The output of these tests are approved Pilot Vehicle Interface (e_{14}) for F-16.

The relations between stakeholders and activities are below:

$$REQUIRE(a_1(\,)) = \{e_{10}\}$$

$$REQUIRE(a_2(\,)) = \{e_{11}\}$$

$$REQUIRE(a_3(\,)) = \{e_{11}\}$$

$$REQUIRE(a_7(\,)) = \{e_{10}\}$$

$$REQUIRE(a_8(\,))=\{e_{10}\}$$

$$REQUIRE(a_4(\,))=\{e_{12},e_{15}\}$$

$$REQUIRE(a_5(\,))=\{e_{13}\}$$

$$REQUIRE(a_6(\,))=\{e_{13}\}$$

The Require relation determines the stakeholders who will perform or participate in any activity. General Dynamics Project Team (e_{10}) is the performer of activities a_1, a_7 and a_8. The General Dynamics Avionic Department is responsible for activities a_2, and a_3. The Cockpit re-mechanization proposal is prepared by Cockpit Review Team (e_{12}), and approved by System Program Office (e_{15}). TAC team (e_{13}) is responsible for designing (a_5) and testing (a_6) the new cockpit layout for F-16 aircraft.

The mathematical model of the project comprising activities, inputs, outputs, stakeholders and relations between these is completed above. Figure 32 shows the graphical model of the F-16 project.

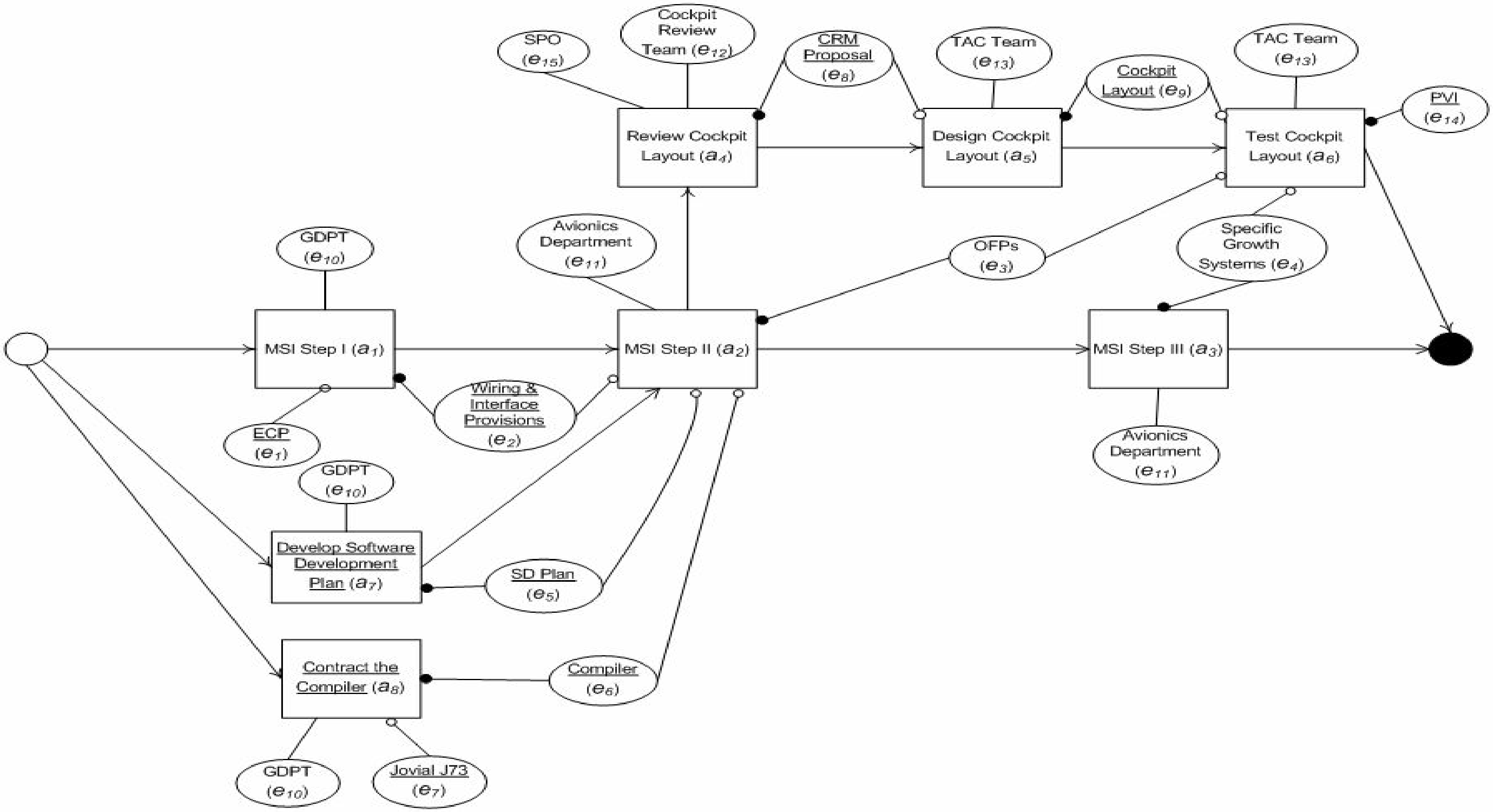

SPO (e_{15})
Cockpit Review Team (e_{12})
CRM Proposal (e_8)
TAC Team (e_{13})
Cockpit Layout (e_9)
TAC Team (e_{13})
PVI (e_{14})
Review Cockpit Layout (a_4)
Design Cockpit Layout (a_5)
Test Cockpit Layout (a_6)
GDPT (e_{10})
Avionics Department (e_{11})
OFPs (e_3)
Specific Growth Systems (e_4)
MSI Step I (a_1)
MSI Step II (a_2)
MSI Step III (a_3)
Wiring & Interface Provisions (e_2)
ECP (e_1)
Avionics Department (e_{11})
GDPT (e_{10})
Develop Software Development Plan (a_7)
SD Plan (e_5)
Contract the Compiler (a_8)
Compiler (e_6)
GDPT (e_{10})
Jovial J73 (e_7)

Figure 32. F-16 Project Highest Level Software Development Plan

The graphical model comprises entire activities, inputs, outputs and relations between them. The graphical model represents the mathematical model. All the relations are represented in the model.

Developing the software plan and contracting the compiler activities are underlined and they are not detailed. They do not have sub-activities or tasks. They have textual explanations determining how they are to be performed. All other activities have sub-activities and tasks that require further modeling.

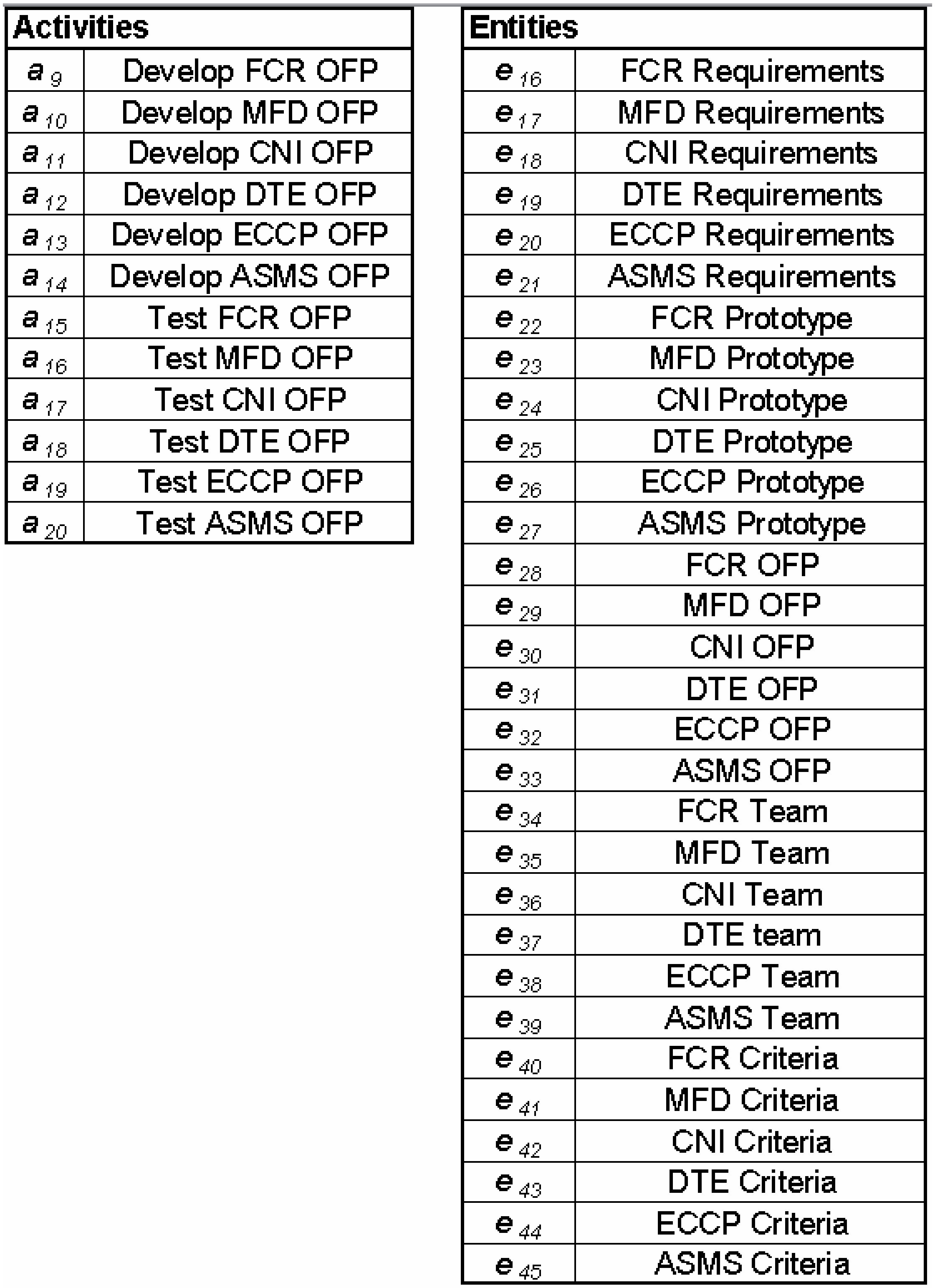
Ellipses connected to activities with a line ending with a blank circle at the activity depict inputs to activities. They represent rules, regulations, physical entities that performers of any activity should address while executing their job. Ellipses connected to activities with a line starting at the activity with a black circle represent outputs or deliverables of activities. Stakeholders and other entities that are required for the activities are depicted with ellipses connected to the activities with solid lines.

Convincing the owners and motivating the performers of the project is an important consideration for the project manager. The graphical model could ease the project managers' job to sell the project to the stakeholders. Anybody participating in the project including the owners can follow what is going on with the project from the model. Stakeholders can also follow their responsibilities and what is expected from them. The Model is a valuable aid to performers of the project.

B. MULTISTAGE IMPROVEMENT PLAN STAGE II

As mentioned above, MSIP Stage II has sub-activities and entities. We will continue our case study with further diving into the sub-activities and entities of MSIP Stage II. MSIP Stage II comprises development of Operational Flight Programs for new avionic systems of F-16 for superior war-fighting performance.

Activities and entities of the phase are below:



Activities	
a_{9}	Develop FCR OFP
a_{10}	Develop MFD OFP
a_{11}	Develop CNI OFP
a_{12}	Develop DTE OFP
a_{13}	Develop ECCP OFP
a_{14}	Develop ASMS OFP
a_{15}	Test FCR OFP
a_{16}	Test MFD OFP
a_{17}	Test CNI OFP
a_{18}	Test DTE OFP
a_{19}	Test ECCP OFP
a_{20}	Test ASMS OFP

Entities	
e_{16}	FCR Requirements
e_{17}	MFD Requirements
e_{18}	CNI Requirements
e_{19}	DTE Requirements
e_{20}	ECCP Requirements
e_{21}	ASMS Requirements
e_{22}	FCR Prototype
e_{23}	MFD Prototype
e_{24}	CNI Prototype
e_{25}	DTE Prototype
e_{26}	ECCP Prototype
e_{27}	ASMS Prototype
e_{28}	FCR OFP
e_{29}	MFD OFP
e_{30}	CNI OFP
e_{31}	DTE OFP
e_{32}	ECCP OFP
e_{33}	ASMS OFP
e_{34}	FCR Team
e_{35}	MFD Team
e_{36}	CNI Team
e_{37}	DTE team
e_{38}	ECCP Team
e_{39}	ASMS Team
e_{40}	FCR Criteria
e_{41}	MFD Criteria
e_{42}	CNI Criteria
e_{43}	DTE Criteria
e_{44}	ECCP Criteria
e_{45}	ASMS Criteria

Table 9. MSIP Stage II Activity / Entity List

In addition to the activities and entities above there are decisions regarding the results of test activities. If the results are satisfactory, the decision is to end the phase; if not, the decision is to return to development activity.

Decisions			
decision$_1$	**Approve FCR OFP**	**decision$_7$**	**Reject FCR OFP**
decision$_2$	**Approve MFD OFP**	**decision$_8$**	**Reject MFD OFP**
decision$_3$	**Approve CNI OFP**	**decision$_9$**	**Reject CNI OFP**
decision$_4$	**Approve DTE OFP**	**decision$_{10}$**	**Reject DTE OFP**
decision$_5$	**Approve ECCP OFP**	**decision$_{11}$**	**Reject ECCP OFP**
decision$_6$	**Approve ASMS OFP**	**decision$_{12}$**	**Reject ASMS OFP**

Table 10. MSIP Stage II Decisions Table

MSIP Stage II relation to the highest level comprising project management issues is:

$$\begin{aligned}DIVIDE(a_2(\,))=\{&start_{a_2},a_9(),a_{10}(),a_{11}(),a_{12}(),a_{13}(),a_{14}(),a_{15}(),a_{16}(),a_{17}(),a_{18}(),a_{19}(),\\ &a_{20}(),e_{16},e_{17},e_{18},e_{19},e_{20},e_{21},e_{22},e_{23},e_{24},e_{25},e_{26},e_{27},e_{28},e_{29},e_{30},\\ &e_{31},e_{32},e_{33},e_{34},e_{35},e_{36},e_{37},e_{38},e_{39},e_{40},e_{41},e_{42},e_{43},e_{44},e_{45},end_{a_2}\}\end{aligned}$$

The relation above includes entire activities, inputs, outputs and stakeholders in MSIP Stage II. MSIP Stage II (a_2) comprises twelve activities from a_9 to a_{20}, twenty-nine entities form e_{16} to e_{45}.

Relations between activities are as follows:

$$start_{a_2} \rightarrow a_9(\,)$$

$$start_{a_2} \rightarrow a_{10}(\,)$$

$$start_{a_2} \rightarrow a_{11}(\,)$$

$$start_{a_2} \rightarrow a_{12}(\,)$$

$$start_{a_2} \rightarrow a_{13}(\,)$$

$$start_{a_2} \rightarrow a_{14}(\,)$$

$$a_9(\,) \rightarrow a_{15}(\,)$$

$$a_{10}(\,) \rightarrow a_{16}(\,)$$

$$a_{11}(\,) \rightarrow a_{17}(\,)$$

$$a_{12}(\,) \rightarrow a_{18}(\,)$$

$$a_{13}(\,) \rightarrow a_{19}(\,)$$

$$a_{14}(\,) \rightarrow a_{20}(\,)$$

$$DECISION(e_{28}, e_{40}) = \left\{ \begin{array}{l} \{decision_1, end_{a_2}\}, \\ \{decision_7, a_9\} \end{array} \right\}$$

$$DECISION(e_{29}, e_{41}) = \left\{ \begin{array}{l} \{decision_2, end_{a_2}\}, \\ \{decision_8, a_{10}\} \end{array} \right\}$$

$$DECISION(e_{30}, e_{42}) = \left\{ \begin{array}{l} \{decision_3, end_{a_2}\}, \\ \{decision_9, a_{11}\} \end{array} \right\}$$

$$DECISION(e_{31}, e_{43}) = \left\{ \begin{array}{l} \{decision_4, end_{a_2}\}, \\ \{decision_{10}, a_{12}\} \end{array} \right\}$$

$$DECISION(e_{32},e_{44})=\begin{Bmatrix}\left\{decision_5,end_{a_2}\right\}, \\ \left\{decision_{11},a_{13}\right\}\end{Bmatrix}$$

$$DECISION(e_{33},e_{45})=\begin{Bmatrix}\left\{decision_6,end_{a_2}\right\}, \\ \left\{decision_{12},a_{14}\right\}\end{Bmatrix}$$

MSIP Stage II comprises six parallel phases independent of each other. Each phase comprises a development activity (a_9 to a_{14}) and a test activity (a_{15} to a_{20}). The results of the test activities require a decision to end the phase or return to development for further corrections and improvements. If the requirements are not satisfied after the test, the decision is to return to development activities (a_9 to a_{14}). If the test results satisfy the requirements, the phase ends.

Relations between activities, inputs and outputs are as follows:

$$e_{22}=a_9(e_{16})$$

$$e_{23}=a_{10}(e_{17})$$

$$e_{24}=a_{11}(e_{18})$$

$$e_{25}=a_{12}(e_{19})$$

$$e_{26}=a_{13}(e_{20})$$

$$e_{27}=a_{14}(e_{21})$$

$$e_{28}=a_{15}(e_{22})$$

$$e_{29}=a_{16}(e_{23})$$

$$e_{30}=a_{17}(e_{24})$$

$$e_{31} = a_{18}(e_{25})$$

$$e_{31} = a_{19}(e_{26})$$

$$e_{33} = a_{20}(e_{27})$$

Each of six parallel development phases have their own requirements regarding the system that will be developed. These requirements (e_{16} to e_{21}) are inputs to development tasks of the systems (a_9 to a_{10}). Outputs or deliverables of the activities are Operational flight programs (e_{28} to e_{33}) for the approval of System Program Office.

The relationships between stakeholders and activities are below:

$$REQUIRE(a_9) = \{e_{34}\}$$

$$REQUIRE(a_{10}) = \{e_{35}\}$$

$$REQUIRE(a_{11}) = \{e_{36}\}$$

$$REQUIRE(a_{12}) = \{e_{37}\}$$

$$REQUIRE(a_{13}) = \{e_{38}\}$$

$$REQUIRE(a_{14}) = \{e_{39}\}$$

$$REQUIRE(a_{16}) = \{e_{15}, e_{34}\}$$

$$REQUIRE(a_{16}) = \{e_{15}, e_{35}\}$$

$$REQUIRE(a_{17}) = \{e_{15}, e_{36}\}$$

$$REQUIRE(a_{18}) = R\{e_{15}, e_{37}\}$$

$$REQUIRE(a_{19}) = \{e_{15}, e_{38}\}$$

$$REQUIRE(a_{20}) = \{e_{15}, e_{39}\}$$

The avionics department assigned six separate teams for the project. Each of the six parallel phases is the responsibility of a separate development team inside the General Dynamics Avionics Department. Development teams (e_{34} to e_{39}) develop and test the software. If the test results are not satisfactory, the task starts from the beginning. The cycle goes on until the requirements are met. If the requirements are satisfied, the System Program Office (e_{11}) approves the final software and the tasks end.

The formulation of MSIP Stage II comprising activities, inputs, outputs, stakeholders and relations between these is completed above. The figure below is the graphical model of the MSIP Stage II based on the relations identified:

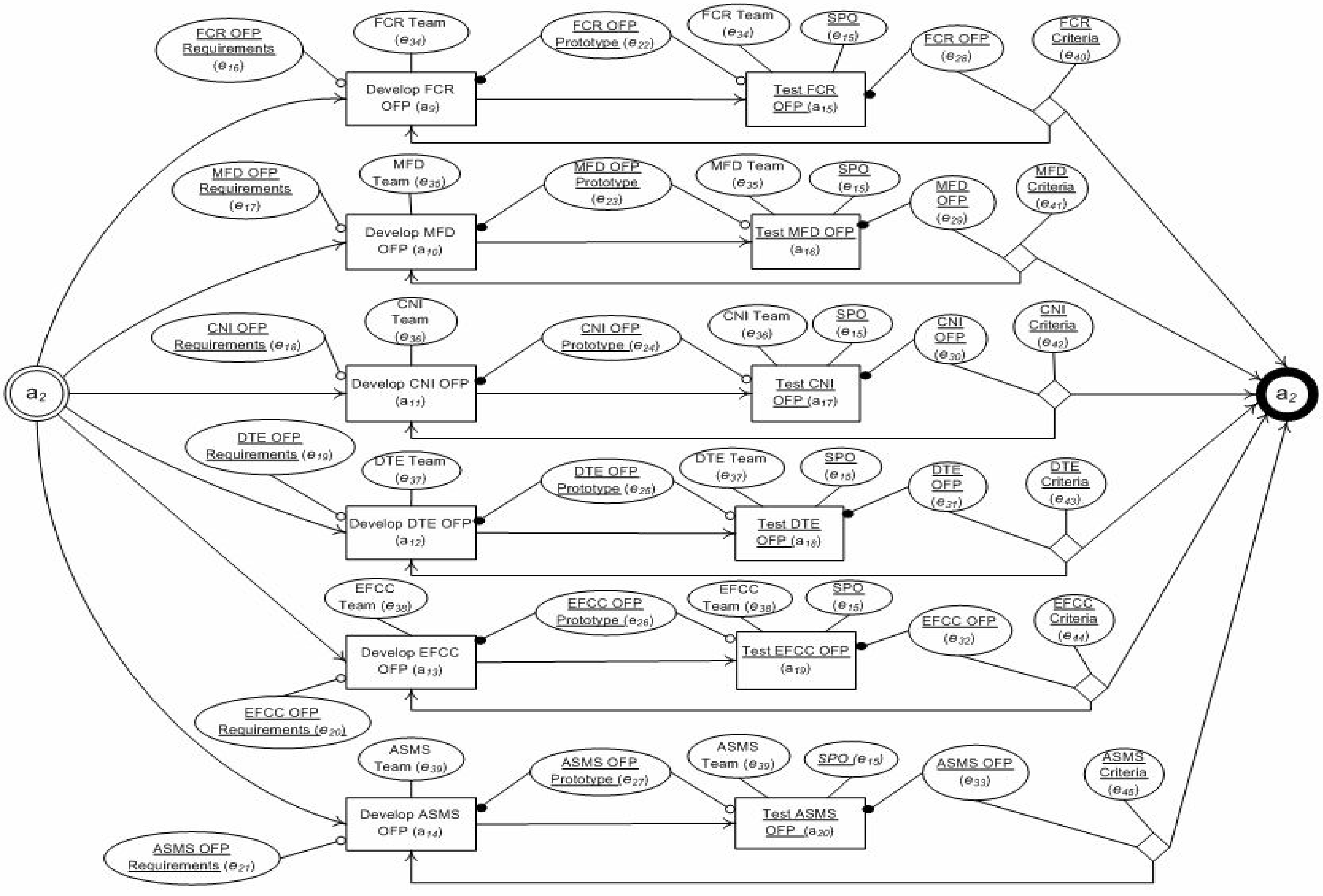


Figure 33. Multistage Improvement Plan Stage II

Development activities and stakeholders of development activities are not underlined in the graphical model. This shows that they have sub-activities and entities. These sub-activities and entities require further modeling. All other activities and entities are underlined. At this point, underlined activities and entities are final and they may have textual explanations. Textual explanations of the underlined activities determine how they will be performed. Textual explanations of inputs determine the requirements for the activities and outputs determine what expected form the model is.

C. EVALUATION OF THE MODEL OF THE CASE STUDY

1. Simplicity

The formulas and model are quite simple. They are composed of relatively few different formulas and shapes. They are easy to understand and follow.

2. Inclusion of Different Aspects of the Software Projects

The model covers scope and quality by displaying inputs and outputs. Inputs and outputs determine what are required in what quality. The model covers people by displaying stakeholders and their relations to activities. However, the model does not cover cost and budget issues of the project. Since the model covers all of the activities and their sequence of execution, schedule can be added to the model with some modification.

3. Work Breakdown Structure of the Project

The model comprises all of the activities, tasks, subtasks of the project. The model also depicts who will perform or participate in any activity. Furthermore, the model includes inputs and outputs of any activity. In essence, project team can determine the work breakdown structure of the project from the model.

4. Inputs and Outputs of the Project

The model comprises all of the inputs and outputs of the project.

5. The Modeling Tool Permits Formal Analysis

The project manager can follow the entire project and derive results from the model. The project manager can follow what modifications, additions or omissions the project requires.

6. Stakeholders and Their Responsibilities and Relations to Any Activity

The model comprises all of the stakeholders participating in the project, their relations to activities.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. CONCLUSION

Software project management is an emerging discipline. A project manager has to deal with foreseen and unforeseen problems during the lifecycle of a project. He or she has to combine both managerial and technical skills to deal with a wide range of issues and improve effective remedies for the project to reach a successful end [23]. In this thesis, we discussed a new theory and modeling language that aims to help to the software project managers' job in planning, executing, controlling, etc. the project.

A. CONCLUSIONS

The proposed theory is relatively simple. It comprises a small amount of shapes and connections. Stakeholders including the project manager and the project team benefit from this simplicity. Individuals do not need a huge amount of time to understand the concept of the theory nor to learn the theory and the modeling language. This simplicity reduces the amount of complexity that a project manager has to deal with. Since the models are simple and easy to understand, the theory helps the project manager in stakeholder buy-in. Besides being simple, the theory also comprises a good amount of different aspects of project management. This makes the theory simple but very useful for a project manager in execution and control of the project. However, this simplicity depends on how successfully the project team models the project.

As determined in Chapter III, it is a desired feature that a new theory would include all aspects of the software project. The theory discussed in this thesis includes all of the stakeholders performing and participating in the project, all of the activities including sub-activities, all of the inputs and deliverables of activities and relations between these aspects. However, the theory excludes cost and schedule aspects of projects.

The models derived by the theory from the case studies, include all of the activities and sub-activities of the projects and the relations between them. WBS is a hierarchical decomposition of the tasks of a project in to sub-tasks. Since models comprise the entire activities of the project, the WBS of the project can be derived form the models. Besides the WBS, an organization chart for the project depicting the personnel and their roles can be derived from the models; however, hierarchic structure of the organization cannot be derived.

Inputs are entities or specifications that are required to complete a task. Deliverables or desired products of activities are outputs of the project. The modeling tool comprises all of the inputs to any activity and desired outputs of the activities. This gives project team a chance to define and control the required inputs of the activities and to define and test the desired outputs of the activities.

The individuals examining the project can derive results from the models. The models help to analyze the activities and entities of the project. All the relations and dependencies can be determined out from the models. These features set a baseline for formal analysis of the projects.

The models derived from the theory deal with four aspects of the software projects: activities, inputs, outputs, stakeholders and relations between those. It does not include cost and schedule. The theory and the modeling language require some extensions or modifications to comprise cost and schedule aspects of software project management.

The mathematical formulation and the graphical models derived from the theory involve all of the stakeholders participating in the software project. Theory defines the stakeholders and their relations to any activity. This feature might help in stakeholder buy-in. Stakeholders can easily see the activities they are expected to participate in or perform.

B. RECOMMENDATIONS

The theory provides a comprehensive tool to the project manager. However, schedule, cost, and risk management are some areas that the theory does not address yet. Future enhancements should focus on integrating new futures to the model that addresses these areas.

Those areas can be addressed with enhancements to the mathematical formulation. The existing mathematical formulation and the graphical model comprises all of the activities, inputs, outputs and stakeholders of a project. Cost can be added to the model by assigning pre-estimated values to all of these activities and entities. Estimation of those values requires usage of a proven cost estimation methodology like COCOMO (**Co**nstructive **Co**st **Mo**del).

Mathematical formulation and models developed from the theory involves the sequence of activities, whether they are parallel or consecutive. Schedule can be added to the model by assigning estimated dates to activities. With this addition, a project schedule can be derived from the models.

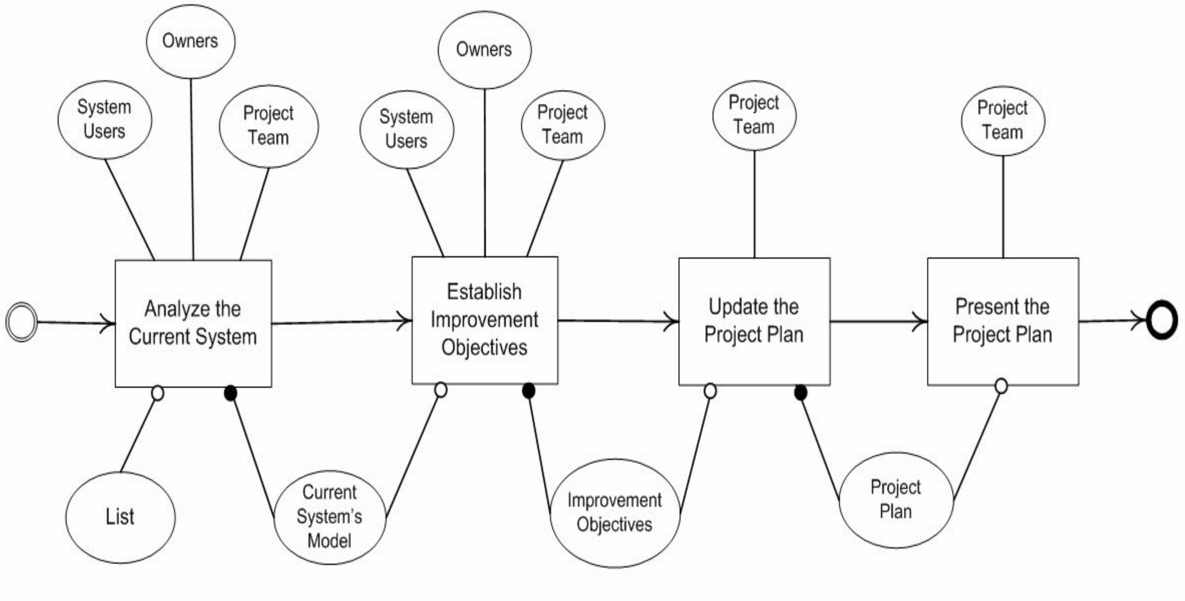
Adding cost and schedule to the models is a valuable enhancement to the theory. However, displaying everything on a single model brings complexity to the theory. Separate models can be built to display cost and schedule aspects of the software projects.

Risk management is about controlling the project risk in an acceptable level. Total risk is defined by assigning risk values to activities and entities. It is project manager’s job to define project risk and control the project not to exceed acceptable risk limits. Since it is a responsibility of the project manager and the project team, adding a risk model to the theory could be a valuable contribution to the theory.

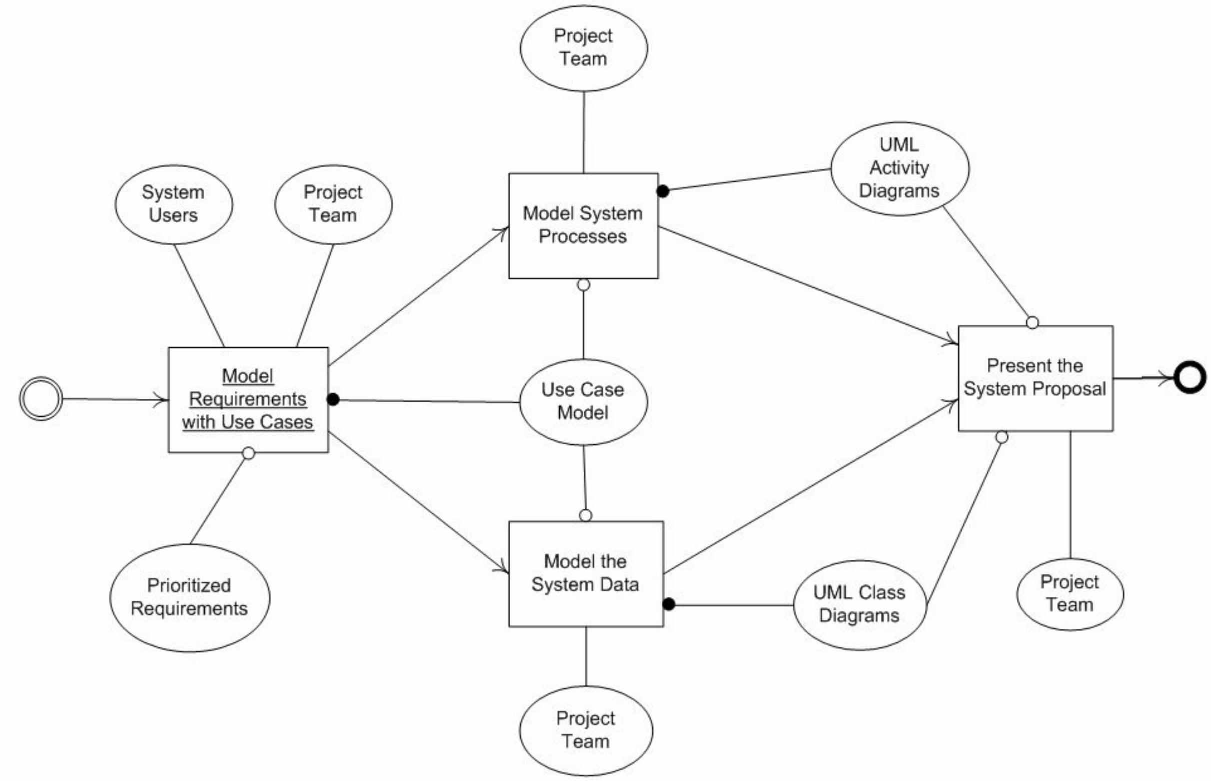
In this thesis, we tested the theory and the modeling language on two case studies. The results of these case studies indicate that the new theory and the modeling language show promise as a valuable tool for project managers.

However, every project comprises some unique feature that is not tested in this thesis. Future applications of the theory to other real life examples can help to determine the missing aspects in the theory.

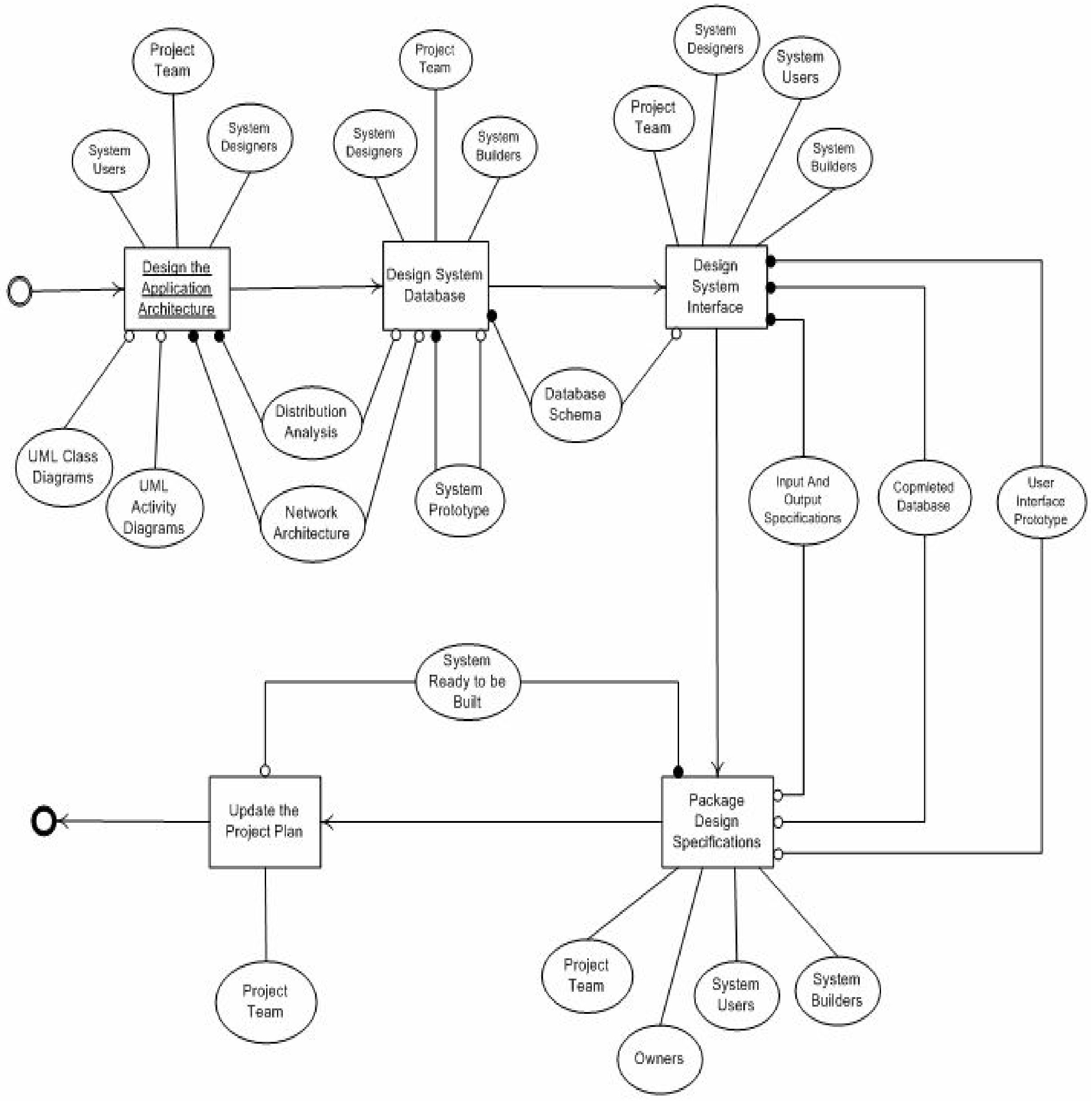
APPENDIX. SOUNDSTAGE PROJECT MODELS



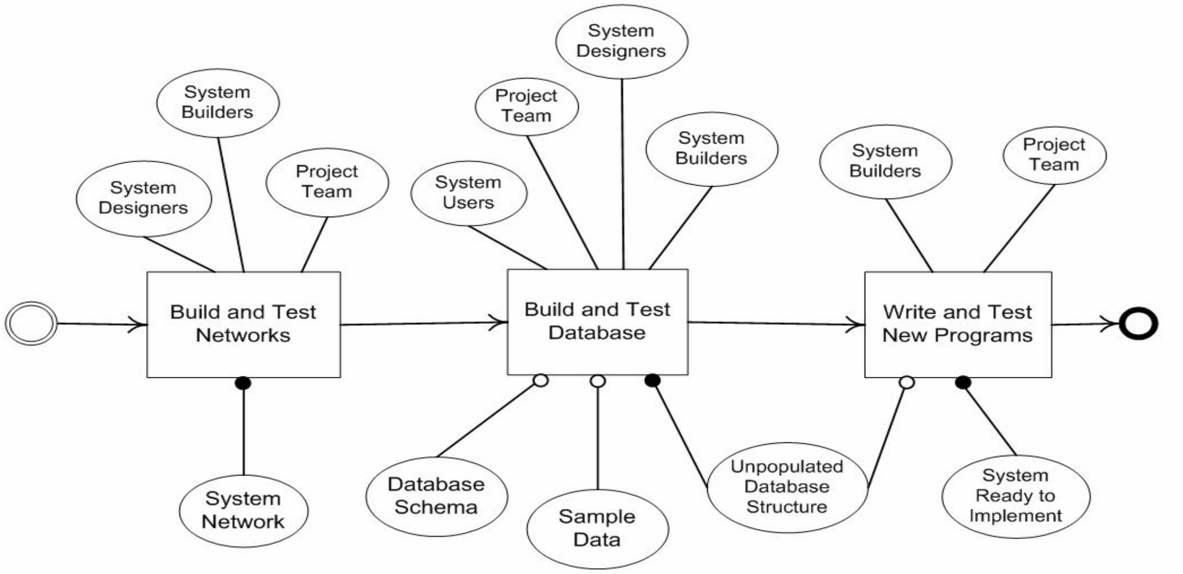
Sound Stage Entertainment Club Problem Analysis Phase



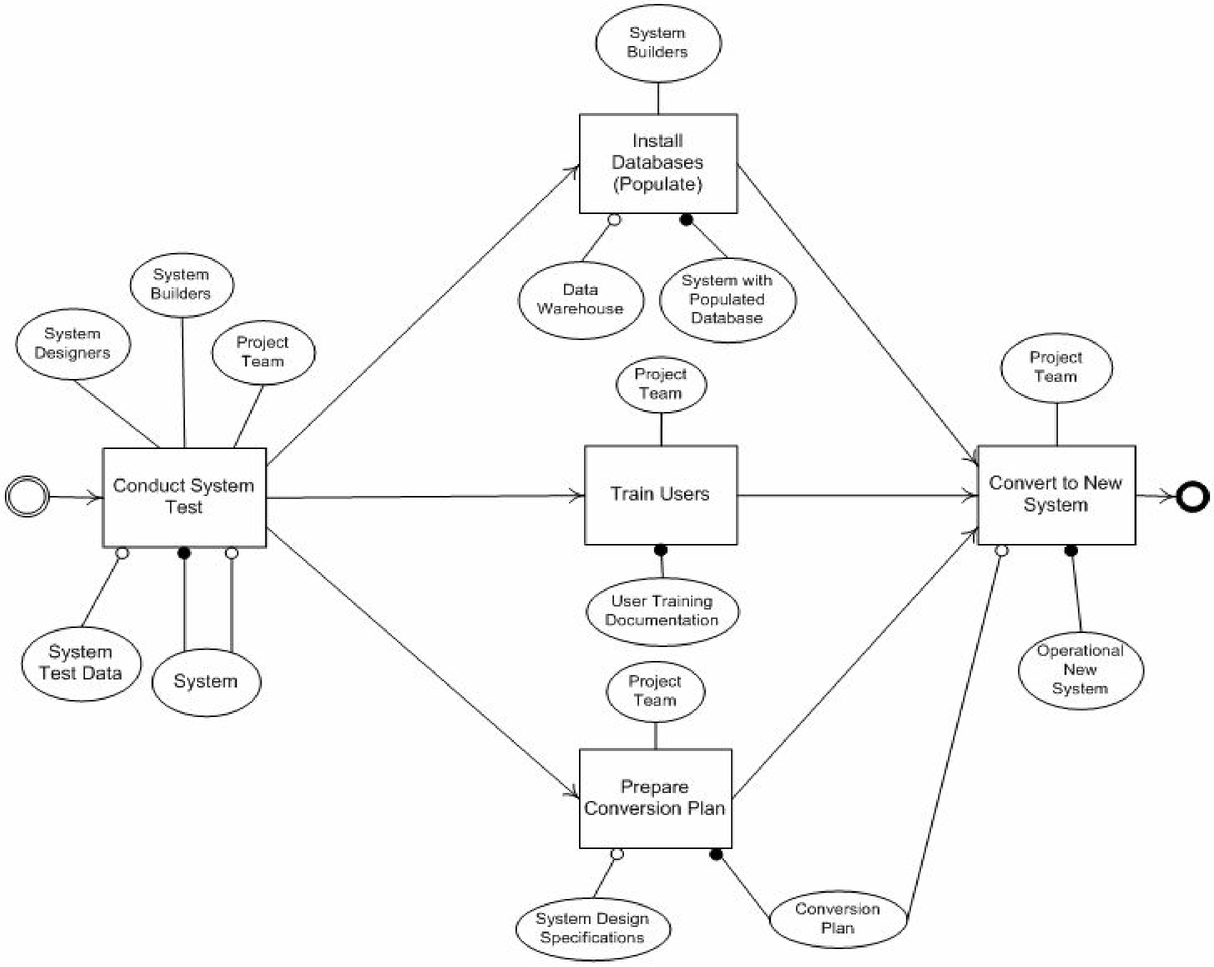
Sound Stage Entertainment Club Logical Design Phase



Sound Stage Entertainment Club Physical Design Phase



Sound Stage Entertainment Club Construction and Testing Phase



Sound Stage Entertainment Club Implementation Phase

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] Project Management Institute, *A Guide to the Project Management Body of Knowledge (PMBOK Guide).* Pennsylvania: Project Management Institute Inc., 2000.

[2] R. L. Kleim and I. S. Ludin, *Project Management Practitioner's Handbook.* New York: Amacom Books, 1998.

[3] L. Vella, "Value of Project Management," *www.gtislig.org/Documents/Vella_PMI-GTISLIG_Value_Of_PMv9. pdf;* Accessed December 2007.

[4] J. Schmitt, "PMO or the Status Quo," *www.jdresources.com/Flash-PDF/PMO_or_the_Status_Quo.pdf*; Accessed December 2007.

[5] J. E. Tomayko and H. K Hallman, "Software Project Management," 1989, ftp://ftp.sei.cmu.edu/pub/documents/misc/cms/pdf//cm21.pdf, Accessed November 2007.

[6] J. Musser, "Principles of Project Management Class Notes," http://www.projectreference.com, Accessed November 2007.

[7] M. G. Lee, P. Yu, and T. L. Jefferson, "Eclectic Software Development Methodology and successful software development", in *IASTED Conf. on Software Engineering*, Innsbruck, Austria, 2005.

[8] "Software project management," class notes for IS4300, Department of Information Sciences, Naval Post Graduate School, Fall 2006.

[9] IBM Rational Software, "Rational Unified Process: Best Practices for Software Development Teams," 1998, *http://www.ibm.com/developerworks/rational/library/253.html*, Accessed January 2008.

[10] R. Jeffries, "What is Extreme Programming," *http://www.xprogramming.com/xpmag/whatisxp.htm*, Accessed December 2007.

[11] K. A. Demir, "Software project management metric," Ph.D. dissertation proposal, Naval Postgraduate School, Monterey, CA, 2005.

[12] M. W. Newell and M. N. Grashina, *The Project Management Question and Answer Book.* New York: Amacom Books, 2004.

[13] S. Bennet, J. Skelton, and K. Lunn, *UML.* Vincenza: McGraw-Hill, 2001.

[14] J. Osmundson and T. V. Hunyh, “A systems engineering methodology for analyzing System of Systems using the Systems Modeling Language (SysML),” in *2nd Annual System of Systems Engineering Conference*, 2006.

[15] Microsoft Corporation, “Microsoft Office Project 2007,” *http://office.microsoft.com/en-us/project*, Accessed January 2008.

[16] Projity Inc, “OpenProj,” http://www.openproj.org/openproj, Accessed February 2008.

[17] Attask Inc, “Product Overview,” *http://www.attask.com /overview*, Accessed February 2008.

[18] toptenREVIEWS Inc., “Project Management Software Review,” *http://project-management-software-review.toptenreviews. com*, Accessed February 2008.

[19] Project.net, “The Project.net Experience,” http://www.project.net, Accessed January 2008.

[20] J. Glatstein, “Formal Visual Analysis: The Elements & Principles of Composition,” *http://artsedge.kennedy-center.org/content/3902*, Accessed January 2008.

[21] K. A. Demir, J. Osmundson, and A. Erguner, “A novel project management modeling tool,” Submitted for Review to *Information & Management*, March 2008.

[22] J. L. Whitten, L. D. Bentley, and K.C. Dittamn, *System Analysis and Design Methods.* 6th ed., New York: McGraw-Hill, 2004.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Chairman
   Information Sciences Department
   Monterey, California

4. Professor John Osmundson
   Department of Information Sciences
   Monterey, California

5. Kadir A. Demir
   Department of Information Sciences
   Monterey, California

6. Abdulkerim Erguner
   Turkish Air Force
   Turkey

7. Hava Kuvvetleri Komutanligi
   Hava Kuvvetleri Komutanligi Kutuphanesi
   Bakanliklar, Ankara, Turkey

8. Hava Harp Okulu
   Hava Harp Okulu Kutuphanesi
   Yesilyurt, Istanbul, Turkey

9. Kara Harp Okulu
   Kara Harp Okulu Kutuphanesi
   Bakanliklar, Ankara, Turkey

10. Deniz Harp Okulu
    Deniz Harp Okulu Kutuphanesi
    Tuzla, Istanbul, Turkey